高等学校计算机专业规划教材

Android程序设计



范永开 许林 编著

清华大学出版社

高等学校计算机专业规划教材

Android 程序设计

范永开　许　林　编著

清华大学出版社
北　京

内 容 简 介

随着移动应用的普及，Android 程序设计演变成高等院校计算机学科的一门基础课程，许多高校将其列入必修或选修课环节。

本书以基础知识结合实际案例的方式，由浅入深地讲解 Android 开发技术。本书采用多例子式论述展开，考虑高等院校的教学需求，对 Android 程序设计中的核心知识点进行说明，通过简单示例学习重要知识点。全书分为 10 章，内容包括 Android 程序设计的基础知识、Android 界面控件的介绍、Android 事件的详细阐述、Android 程序设计的界面布局、Android 的弹出信息与资源、Android 数据存储与网络应用。同时，对书中内容以程序实例的方式进行阐述，语言通俗易懂，示例丰富实用，能帮助读者拨开晦涩难懂的术语迷雾，一步一步地进行详细指导式学习。

本书从书写与内容设计方面着重考虑适合作为 64 学时的高等院校理工类学生的教材，同时也可作为 Android 程序开发者的参考书。

图书在版编目（CIP）数据

Android 程序设计/范永开，许林编著. --北京：清华大学出版社，2014
高等学校计算机专业规划教材
ISBN 978-7-302-34502-2

Ⅰ. ①A… Ⅱ. ①范… ②许… Ⅲ. ①移动终端—应用程序—程序设计—高等学校—教材
Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字(2013)第 274260 号

责任编辑：龙启铭　王冰飞
封面设计：
责任校对：时翠兰
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：16　　**字　　数**：367 千字
版　　次：2014 年 6 月第 1 版　　**印　　次**：2014 年 6 月第1 次印刷
印　　数：1～　000
定　　价：　.00 元

产品编号：053510-01



前言

自从 Google 公司在 2007 年推出 Android 以来，Android 已经经历了 6 年多的发展。从版本 1.5 开始，Android 崭露头角，也开始进入了高速发展阶段。基于 Android 的智能设备是移动领域最具有人气的设备之一。Android 以免费、开源为特点，并且 Google 没有限制使用什么语言或技术在 Android 上开发软件。这就意味着任何企业、组织和个人都可以使用 Android 系统，这也使 Android 的市场占有率节节攀升。无论在国内还是国外，Android 都受到极大的关注和重用，在可预期的未来，在全球范围内的移动领域，Android 将扮演一个重要的角色。

作为传播知识与教育的核心领域——高等院校顺应时代发展的趋势，紧跟信息领域发展的脉搏，许多高校及时调整自己的培养体系，已将 Android 相关知识作为本科生教育的一部分，这部分内容以必修课或选修课的形式出现在课程体系结构中。本书是作者作为一线教师及一线的开发者近几年来讲课与开发的结晶，书中所出现的代码是作者书写程序近 20 年的个人心得与新技术出现的熔融。本书克服文字说教的方式，通过实例来论述知识点，强调在学习过程中提高编程能力，真正回归语言学习的真谛。

本书旨在帮助读者认识 Android，并从各个角度介绍 Android 的核心理念和学习方法，从多个方面介绍和阐述对 Android 架构的理解。只有真正理解 Android 的设计理念和思想，才能更快地掌握 Android 开发。本书深入阐述了 Android 最核心的控件与事件机制，详细介绍了控件的特点和使用方式，以 Android 的交互界面开发为目的，讲解了 Android 的控件框架，结合实际的项目，对重要控件的实现和使用逐一进行了分析与讲解，其中包含了最新的 Android 界面开发的一些实践精华。本书还阐述了 Android 的数据存储结构，不同的数据存储模式的使用要点以及简单的网络应用。

本书非常适合于 64 学时的高等院校理工类学生学习 Android 程序设计，对于具有丰富 Android 开发经验、对 Android 基础框架有很深认识的资深开发者而言，本书可作为一本参考用书。

本书得到中国石油大学（北京）优秀青年教师基金（2462012KYJJ0518）资助，在此表示感谢！

编　者

2014 年 5 月


目 录

第4章 Android事件 /94

第5章 Android布局 /123

第6章 Android弹出信息 /147

第7章 Android资源 /167

第1章

Android 概 述

当前参与移动业务领域的有四类人群：移动用户、移动运营商、移动开发人员和移动产品制造商。移动用户需要更多的、更实用的、更个性化的功能。移动运营商需要更易于管理、获利更多的增值服务。移动开发人员需要可以自由开发的开发环境，不要有太多限制。移动产品制造商需要一个稳定、安全、廉价的系统平台，在充分发挥硬件的性能的基础上降低成本。Android 系统是唯一可以满足以上四类人群需要的移动开发平台。

未来 IT 行业中，移动业务将占有举足轻重的地位。越来越多的公司，包括电商、金融、餐饮等行业的公司都在实现自己的移动业务系统，以方便客户使用。Android 系统是普遍采用的移动操作系统。

本书所介绍的 Android 系统是基于 Android 4.0 版本的，这是当前移动产品采用比较广泛的平台版本。

作为学习和了解 Android 的第一步，在这一章中，我们将简要地介绍 Android 的由来、发展和优势。

1.1　Android 的由来和发展

1.1.1　Android 的由来

Android 一词最先出现在法国作家利尔亚当在 1886 年发表的科幻小说《未来夏娃》中，作者将外表像人类的机器起名为 Android，如图 1.1 所示。



图 1.1　Android 的 Logo

Google 将其基于 Linux 平台的开源手机操作系统命名为 Android，该平台由操作系统、中间件、用户界面和应用软件组成，并采用图 1.1 作为 Logo。

Android 的 Logo 是由 Ascender 公司设计的，其中的文字使用了 Ascender 公司专门制作的称为 Droid 的字体。Android 是一个全身绿色的机器人，绿色也是 Android 的标志。颜色采用了 PMS 376C 和 RGB 中十六进制的＃A4C639 来绘制，这是 Android 操作系统的品牌象征。有时候，它们还会使用纯文字的。

1.1.2 Android的发展过程

2003 年 10 月,Andy Rubin 等人创建 Android 公司,并组建 Android 团队。

2005 年 8 月 17 日,Google 低调收购了成立仅 22 个月的高科技企业 Android 及其团队。安迪·鲁宾成为 Google 公司工程部副总裁,继续负责 Android 项目。

2007 年 11 月 5 日,Google 公司正式向外界展示了这款名为 Android 的操作系统,并且在这天 Google 宣布建立一个全球性的联盟组织,该组织由 34 家手机制造商、软件开发商、电信运营商以及芯片制造商共同组成,并与 84 家硬件制造商、软件开发商及电信运营商组成开放手持设备联盟(Open Handset Alliance)来共同研发改良 Android 系统,这一联盟将支持 Google 发布的手机操作系统以及应用软件,Google 以 Apache 免费开源许可证的授权方式发布了 Android 的源代码。

2008 年,在 Google I/O 大会上,Google 提出了 Android HAL 架构图,在同年 8 月 18 日,Android 获得了美国联邦通信委员会(FCC)的批准,在 2008 年 9 月,Google 正式发布了 Android 1.0 系统,这也是 Android 系统最早的版本。

2009 年 4 月,Google 正式推出了 Android 1.5 这款手机,从 Android 1.5 版本开始,Google 开始将 Android 的版本以甜品的名字命名,Android 1.5 命名为 Cupcake(纸杯蛋糕)。该系统与 Android 1.0 相比有了很大的改进。

2009 年 9 月,Google 发布了 Android 1.6 的正式版,并且推出了搭载 Android 1.6 正式版的手机 HTC Hero(G3),凭借着出色的外观设计以及全新的 Android 1.6 操作系统,HTC Hero(G3)成为当时全球最受欢迎的手机。Android 1.6 也有一个有趣的甜品名称,它被称为 Donut(甜甜圈)。

2010 年 2 月,Linux 内核开发者 Greg Kroah-Hartman 将 Android 的驱动程序从 Linux 内核"状态树"("staging tree")上除去,从此,Android 与 Linux 开发主流分道扬镳。在同年 5 月,Google 正式发布了 Android 2.2 操作系统。Google 将 Android 2.2 操作系统命名为 Froyo(冻酸奶)。

2010 年 10 月,Google 宣布 Android 系统达到了第一个里程碑,即电子市场上获得官方数字认证的 Android 应用数量已经达到了 10 万个,Android 系统的应用增长非常迅速。在 2010 年 12 月,Google 正式发布了 Android 2.3 操作系统 Gingerbread(姜饼)。

2011 年 1 月,Google 称每日的 Android 设备新用户数量达到了 30 万部,到 2011 年 7 月,这个数字增长到 55 万部,而 Android 系统设备的用户总数达到了 1.35 亿,Android 系统已经成为智能手机领域占有量最高的系统。

2011 年 8 月 2 日,Android 手机已占据全球智能机市场 48%的份额,并在亚太地区市场占据统治地位,终结了 Symbian(塞班系统)的霸主地位,跃居全球第一。

2011 年 9 月,Android 系统的应用数目已经达到 48 万,而在智能手机市场,Android 系统的占有率已经达到 43%,继续排在移动操作系统首位。在本月 19 日,Google 发布了全新的 Android 4.0 操作系统,这款系统被 Google 命名为 Ice Cream Sandwich(冰激凌三明治)。

2012 年 1 月 6 日,Google Android Market 已有 10 万开发者推出超过 40 万活跃的应

用,大多数的应用程序为免费。Android Market 应用商店在新年首周周末突破 40 万基准,距离突破 30 万应用仅 4 个月。在 2011 年早些时候,Android Market 从 20 万增加到 30 万应用也花了 4 个月。

1.2　Android 的体系结构

Android 的系统架构和其操作系统一样,采用了分层的架构,如图 1.2 所示,分为 4 个层,从高层到低层分别是应用程序层、应用程序框架层、系统运行库层和 Linux 核心层。

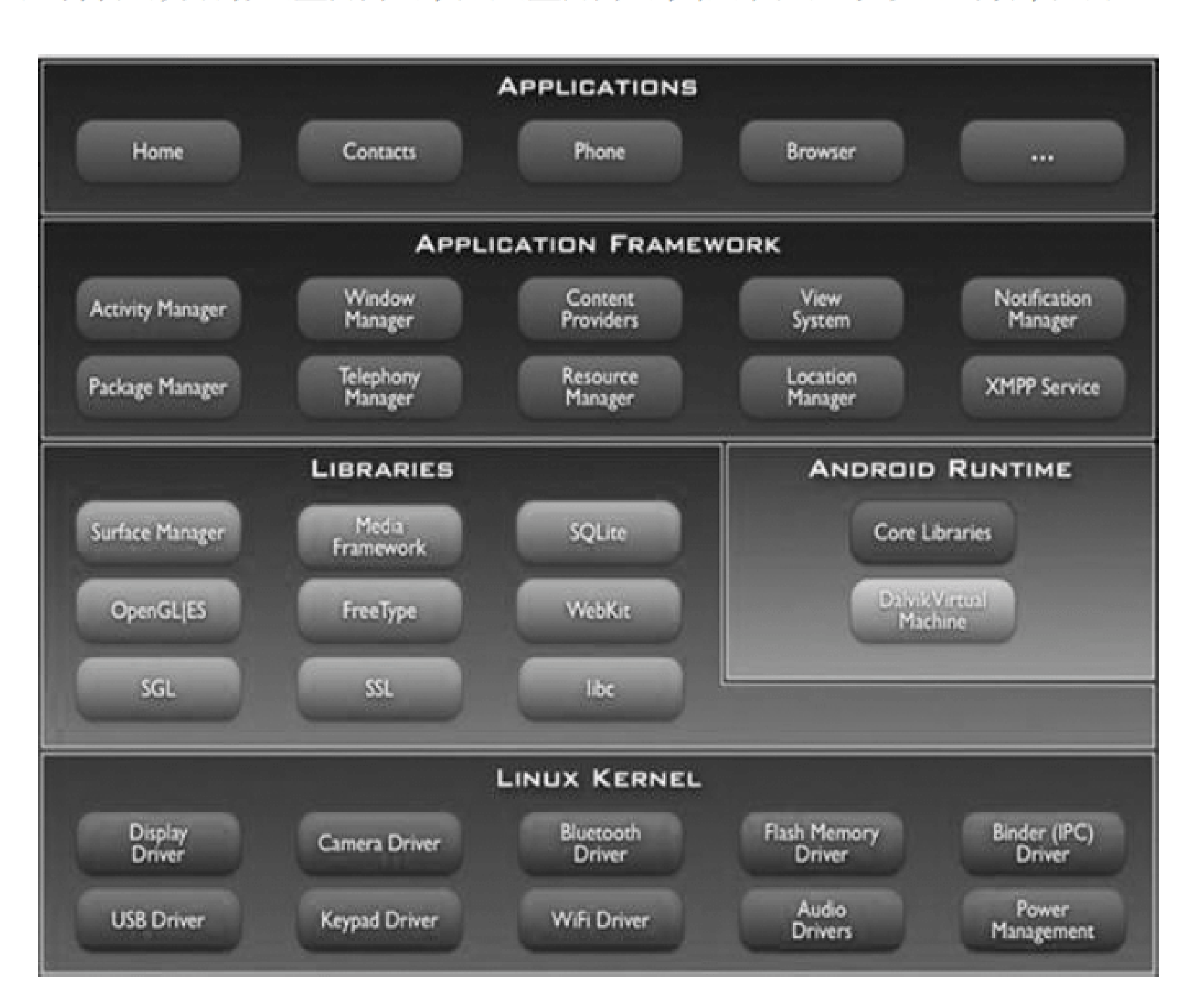


图 1.2　Android 系统架构图

1.2.1　应用程序

Android 会同一系列核心应用程序包一起发布,该应用程序包包括 E-mail 客户端、SMS 短消息程序、日历、地图、浏览器、联系人管理程序等。所有的应用程序都是使用 Java 语言编写的。

1.2.2　应用程序框架

开发人员可以完全访问核心应用程序所使用的 API 框架。该应用程序的架构设计简化了组件的重用;任何一个应用程序都可以发布它的功能块并且任何其他的应用程序都可以使用其所发布的功能块(不过要遵循框架的安全性限制)。同样,该应用程序重用机制也使用户可以方便地替换程序组件。

隐藏在每个应用后面的是一系列的服务和系统,其中包括:

- 丰富而又可扩展的视图(Views),可以用来构建应用程序,它包括列表(lists)、网格(grids)、文本框(text boxes)、按钮(buttons),甚至可嵌入的 Web 浏览器。
- 内容提供器(Content Providers),使得应用程序可以访问另一个应用程序的数据

(如联系人数据库)，或者共享它们自己的数据。

- 资源管理器(Resource Manager)，提供非代码资源的访问，如本地字符串、图形和布局文件(layout files)。
- 通知管理器 (Notification Manager)，使得应用程序可以在状态栏中显示自定义的提示信息。
- 活动管理器(Activity Manager)，用来管理应用程序生命周期并提供常用的导航回退功能。

1.2.3 系统运行库

1. 程序库

Android 包含一些 C/C++ 库，这些库能被 Android 系统中不同的组件使用。它们通过 Android 应用程序框架为开发者提供服务。程序库包括以下核心库。

- 系统 C 库：一个从 BSD 继承来的标准 C 系统函数库(libc)，它是专门为基于嵌入式 Linux 的设备定制的。
- 媒体库：基于 PacketVideoOpenCORE。该库支持多种常用的音频、视频格式回放和录制，同时支持静态图像文件。编码格式包括 MPEG4、H. 264、MP3、AAC、AMR、JPG 和 PNG。
- Surface Manager：对显示子系统的管理，并且为多个应用程序提供了 2D 和 3D 图层的无缝融合。
- LibWebCore：一个最新的 Web 浏览器引擎，支持 Android 浏览器和一个可嵌入的 Web 视图。
- SGL：底层的 2D 图形引擎。
- 3D 库：基于 OpenGL ES 1. 0 APIs 实现。该库可以使用硬件 3D 加速(如果可用)或者使用高度优化的 3D 软加速。
- FreeType：位图(bitmap)和矢量(vector)字体显示。
- SQLite：一个对于所有应用程序可用、功能强劲的轻型关系型数据库引擎。

2. Android 运行库

Android 包括一个核心库，该核心库提供了 Java 编程语言核心库的大多数功能。

每一个 Android 应用程序都在它自己的进程中运行，都拥有一个独立的 Dalvik 虚拟机实例。Dalvik 被设计成一个设备可以同时高效地运行多个虚拟系统。Dalvik 虚拟机执行的 Dalvik 可执行文件(. dex)，该格式文件针对小内存使用做了优化。同时虚拟机是基于寄存器的，所有的类都经由 Java 编译器编译，然后通过 SDK 中的 dx 工具转化成. dex 格式由虚拟机执行。

Dalvik 虚拟机依赖于 Linux 内核的一些功能，比如线程机制和底层内存管理机制。

1.2.4 Linux 内核

Android 的核心系统服务依赖于 Linux 2. 6 内核，如安全性、内存管理、进程管理、网络协议栈和驱动模型。Linux 内核也同时作为硬件和软件栈之间的抽象层。

1. 系统内核

Android 运行于 Linux Kernel 之上,但并不是 GNU/Linux。因为在一般的 GNU/Linux 中支持的功能,Android 大都没有支持,包括 Cairo、X11、Alsa、FFmpeg、GTK、Pango 及 Glibc 等都被移除掉了。Android 又以 Bionic 取代 Glibc,以 Skia 取代 Cairo,再以 opencore 取代 FFmpeg,等等。Android 为了达到商业应用,必须移除被 GNU GPL 授权证所约束的部分,例如 Android 将驱动程序移到 Userspace,使得 Linux 驱动与 Linux 内核彻底分开。Bionic/Libc/Kernel 并非标准的内核头文件。Android 的内核头是利用工具由 Linux 内核头产生的,这样做是为了保留常数、数据结构与宏。

Android 的 Linux 内核控制包括安全(Security)、存储器管理(Memory Management)、程序管理(Process Management)、网络堆栈(Network Stack)和驱动程序模型(Driver Model)等。下载 Android 源码之前,先要安装其构建工具 Repo 来初始化源码。Repo 是 Android 用来辅助 Git 工作的一个工具。

2. 硬件抽象层

Android 的 HAL(硬件抽象层)以封闭源码形式提供硬件驱动模块。HAL 的目的是为了把 Android 框架与 Linux 内核隔开,让 Android 不至于过度依赖 Linux 内核,以达成内核独立的概念,也让 Android 框架的开发能在不考虑驱动程序实现的前提下进行发展。

HAL Stub 是一种代理人(Proxy)的概念,Stub 是以 *. so 档的形式存在的。Stub 向 HAL"提供"操作函数(Operations),并由 Android runtime 向 HAL 取得 Stub 的 Operations,再 Callback 这些操作函数。HAL 中包含了许多的 Stub(代理人)。Runtime 只要说明"类型",即 Module ID,就可以取得操作函数。

3. 中介软件

中介软件是操作系统与应用程序的沟通桥梁,分为两层:函数层(Library)和虚拟机(Virtual Machine)。

Android 采用 OpenCORE 作为基础多媒体框架。Open CORE 可分为七大块:PVPlayer、PVAuthor、Codec、PacketVideo Multimedia Framework(PVMF)、Operating System Compatibility Library(OSCL)、Common 和 OpenMAX。

Android 使用 Skia 作为核心图形引擎,搭配 OpenGL/ES。Skia 与 Linux Cairo 功能相当,但与 Linux Cairo 相比,skia 功能还只是雏形的。2005 年 Skia 公司被 Google 收购,2007 年初,Skia GL 源码被公开,Skia 也是 Google Chrome 的图形引擎。

Android 的多媒体数据库采用 SQLite 数据库系统。数据库又分为共用数据库和私用数据库。用户可通过 ContentResolver 类(Column)取得共用数据库。

Android 的中间层多以 Java 实现,并且采用特殊的 Dalvik 虚拟机(Dalvik Virtual Machine)。Dalvik 虚拟机是一种"暂存器形态"(Register Based)的 Java 虚拟机,变量皆存放于暂存器中,虚拟机的指令相对减少。

Dalvik 虚拟机可以有多个实例(Instance),每个 Android 应用程序都用一个自属的 Dalvik 虚拟机来运行,让系统在运行程序时可达到优化。Dalvik 虚拟机并非运行 Java 字节码(Bytecode),而是运行一种称为.dex 格式的文件。

4. 安全权限机制

Android 本身是一个权限分立的操作系统。在这类操作系统中,每个应用都以唯一

的一个系统识别身份运行(Linux 用户 ID 与群组 ID)。系统的各部分也分别使用各自独立的识别方式。Linux 也是这样将应用与应用、应用与系统隔离开的。

系统更多的安全功能通过权限机制提供。权限可以限制某个特定进程的特定操作，也可以限制每个 URI 权限对特定数据段的访问。

Android 安全架构的核心设计思想是：在默认设置下，所有应用都没有权限对其他应用、系统或用户进行较大影响的操作。这其中包括读写用户隐私数据(联系人或电子邮件)，读写其他应用文件，访问网络或阻止设备待机等。

安装应用时，在检查程序签名提及的权限，且经过用户确认后，软件包安装器会给予应用权限。从用户角度看，一款 Android 应用通常会要求如下的权限：拨打电话、发送短信或彩信、修改/删除 SD 卡上的内容、读取联系人的信息、读取日程信息，写入日程数据、读取电话状态或识别码、精确的(基于 GPS)地理位置、模糊的(基于网络获取)地理位置、创建蓝牙连接、对互联网的完全访问、查看网络状态、查看 WiFi 状态、避免手机待机、修改系统全局设置、读取同步设定、开机自启动、重启其他应用、终止运行中的应用、设定偏好应用、震动控制、拍摄图片等。

一款应用应该根据自身提供的功能，要求合理的权限。用户也可以分析一款应用所需权限，从而简单判定这款应用是否安全。例如，一款应用是不带广告的单机版，也没有任何附加的内容需要下载，那么它要求访问网络的权限就比较可疑。

1.3 Android 的优劣

1.3.1 Android 平台手机的五大优势

1. 开放性

Android 平台的最大优势就是其开放性，Android 平台允许任何移动终端厂商加入到 Android 联盟中来。显著的开放性可以使其拥有更多的开发者，随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟。

开放性对于 Android 的发展而言，有利于积累人气，这里的人气包括消费者和厂商。对于消费者来说，最大的受益是丰富的软件资源。开放的平台也会带来更大竞争，如此一来，消费者将可以用更低的价位购得心仪的手机。

2. 挣脱运营商的束缚

在过去很长的一段时间，特别是在欧美地区，手机应用往往受到运营商的制约，使用什么功能、接入什么网络，几乎都受到运营商的控制。从 iPhone 上市后，用户可以更加方便地连接网络，运营商的制约减少。随着 EDGE、HSDPA 这些 2G 至 3G 移动网络的逐步过渡和提升，手机随意接入网络已不是运营商口中的笑谈，当你通过手机 IM 软件方便地进行即时聊天时，再回想不久前天价的彩信和图铃下载业务，是不是像噩梦一样?

互联网巨头 Google 推动的 Android 终端天生就有网络特色，这让用户离互联网更近。

3. 丰富的硬件选择

这一点还是与 Android 平台的开放性相关，由于 Android 的开放性，众多的厂商会推

出千奇百怪、各具特色的产品。功能上的差异和特色却不会影响到数据同步，甚至软件的兼容，好比从诺基亚 Symbian 风格手机一下改用苹果 iPhone，同时还可将 Symbian 中优秀的软件带到 iPhone 上使用，联系人等资料更是可以方便地转移，是不是非常方便呢？

4. 不受任何限制的开发商

Android 平台提供给第三方开发商一个十分宽泛、自由的环境，不会受到各种条条框框的阻挠，可想而知，会有多少新颖别致的软件诞生。但这个特点也有其两面性，对血腥、暴力、情色方面的程序和游戏如何控制，正是留给 Android 的难题之一。

5. 无缝结合的 Google 应用

如今叱咤互联网的 Google 已经走过 10 多年的历史，从搜索巨人到全面的互联网渗透，Google 服务如地图、邮件、搜索等已经成为连接用户和互联网的重要纽带，而 Android 平台手机将无缝结合这些优秀的 Google 服务。

1.3.2 Android 的五大不足

1. 安全和隐私

由于手机与互联网的紧密联系，个人隐私很难得到保护。除了上网过程中经意或不经意留下的个人足迹，Google 这个巨人也时时站在你的身后，洞穿一切。因此，互联网的深入将会带来新一轮的隐私危机。

2. 首先开卖 Android 手机的不是最大运营商

众所周知，T-Mobile 在 2008 年 9 月 22 日，于美国纽约发布了 Android 首款手机 G1。但是在北美市场，最大的两家运营商是 AT&T 和 Verizon，而目前所知，取得 Android 手机销售权的仅有 T-Mobile 和 Sprint，其中，T-Mobile 的 3G 网络相对于其他三家也要逊色不少，因此，用户可以买账购买 G1，能否体验到最佳的 3G 网络服务则要另当别论了。

3. 运营商仍然能够影响到 Android 手机

在国内市场，不少用户对购得移动定制机不满，感觉所购的手机被人涂画了广告一般。这样的情况在国外市场同样出现。Android 手机的另一发售运营商 Sprint 就将在其机型中内置其手机商店程序。

4. 同类机型用户减少

在不少手机论坛都会有针对某一型号的子论坛，对一款手机的使用交流心得，并分享软件资源。而对于 Android 平台手机，由于厂商丰富，产品类型多样，因此使用同一款机型的用户较少，缺少统一机型的程序强化。举个稍显不当的例子，现在山寨机泛滥，品种各异，就很少有专门针对某个型号山寨机的讨论和群组，除了那些功能异常抢眼、颇受追捧的机型以外。

5. 过分依赖开发商，缺少标准配置

在使用 PC 端的 Windows XP 系统时，都会内置微软 Windows Media Player 这样一个程序，用户还可以选择更多样的播放器，如 RealPlay 或暴风影音等。但入手开始使用默认的程序同样可以应付多样的需要。在 Android 平台中，由于其开放性，软件更多依赖第三方厂商，例如 Android 系统的 SDK 中就没有内置音乐播放器，全部依赖第三方开发，缺少了产品的统一性。

第 2 章

Android 初探

通过第 1 章的学习，我们知道了 Android 的基本情况。下面将指导大家准备 Android 开发环境，并指引大家编写一个简单的 Android 程序以及如何进行简单的测试。

2.1 准备相关软件

下面介绍安装 Android 需要的相关软件。

2.1.1 操作系统

支持 Android 应用程序的操作系统有以下几种：

(1) Windows XP、Windows 7 或 Windows Vista。

(2) Linux。

(3) Mac OS X 10.4.8 及新版本。

2.1.2 JDK

Android 开发环境是基于 Java 的。JDK 是整个 Java 的核心，包括 Java 运行环境、Java 工具和 Java 基础的类库。这是搭建平台所必需的。

JDK 的下载地址是 http://www.oracle.com/technetwork/Java/Javase/downloads/index.html，下载界面如图 2.1 所示。

选择 JDK DOWNLOAD，只下载 JDK，无须下载 JRE。

2.1.3 IDE 开发环境

在 IDE 开发环境中，开发人员多数使用免费的 Eclipse 集成开发环境。Eclipse 的下载地址是 http://www.eclipse.org/downloads/，下载界面如图 2.2 所示。

首先在 Eclipse Juno (4.2) SR1 Packages for 后面的下拉列表框中选择使用的操作系统，然后选择第一个(即 Eclipse IDE for Java EE Developers)下载即可。解压到文件夹之后即可使用，无须安装。

当然，也可以选择 MyEclipse，它是对 Eclipse IDE 的扩展。MyEclipse 的下载地址是 http://www.myeclipseide.com/。下载完成后按照操作提示安装即可。

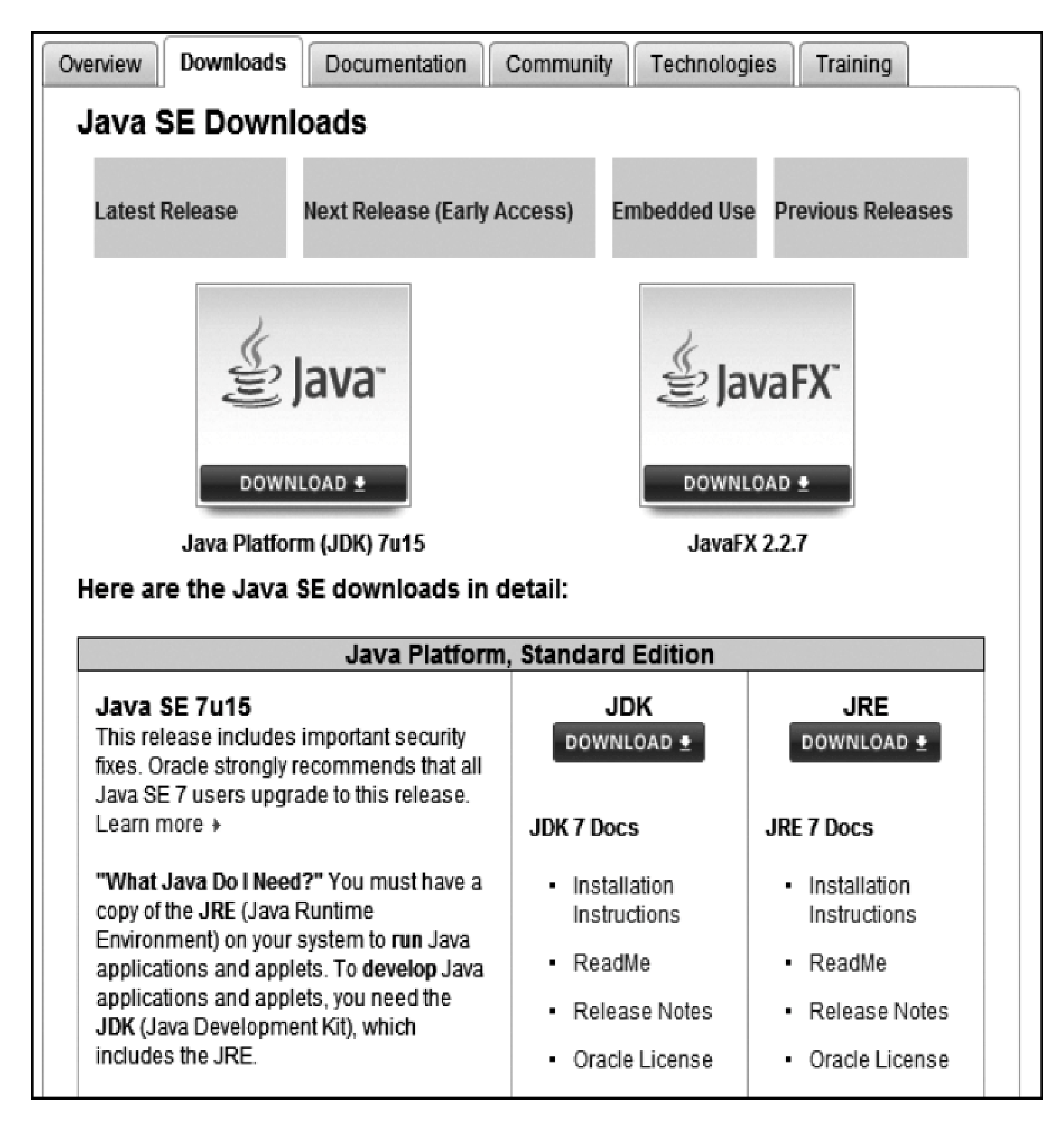


图 2.1 JDK 下载界面



图 2.2 Eclipse 下载界面

2.1.4 软件开发包 Android SDK

SDK 是被软件开发工程师用于为特定的软件包、软件框架、硬件平台、操作系统等建立应用软件的开发工具的集合。

Android SDK 的官方网址是 http://developer.Android.com/sdk/index.html，下载界面如图 2.3 所示。

进入后，跟着流程走，选择适合自己平台的 SDK 包下载。下载后直接解压就可以使用。当前最新的 SDK 软件包包含 Eclipse IDE 开发环境。

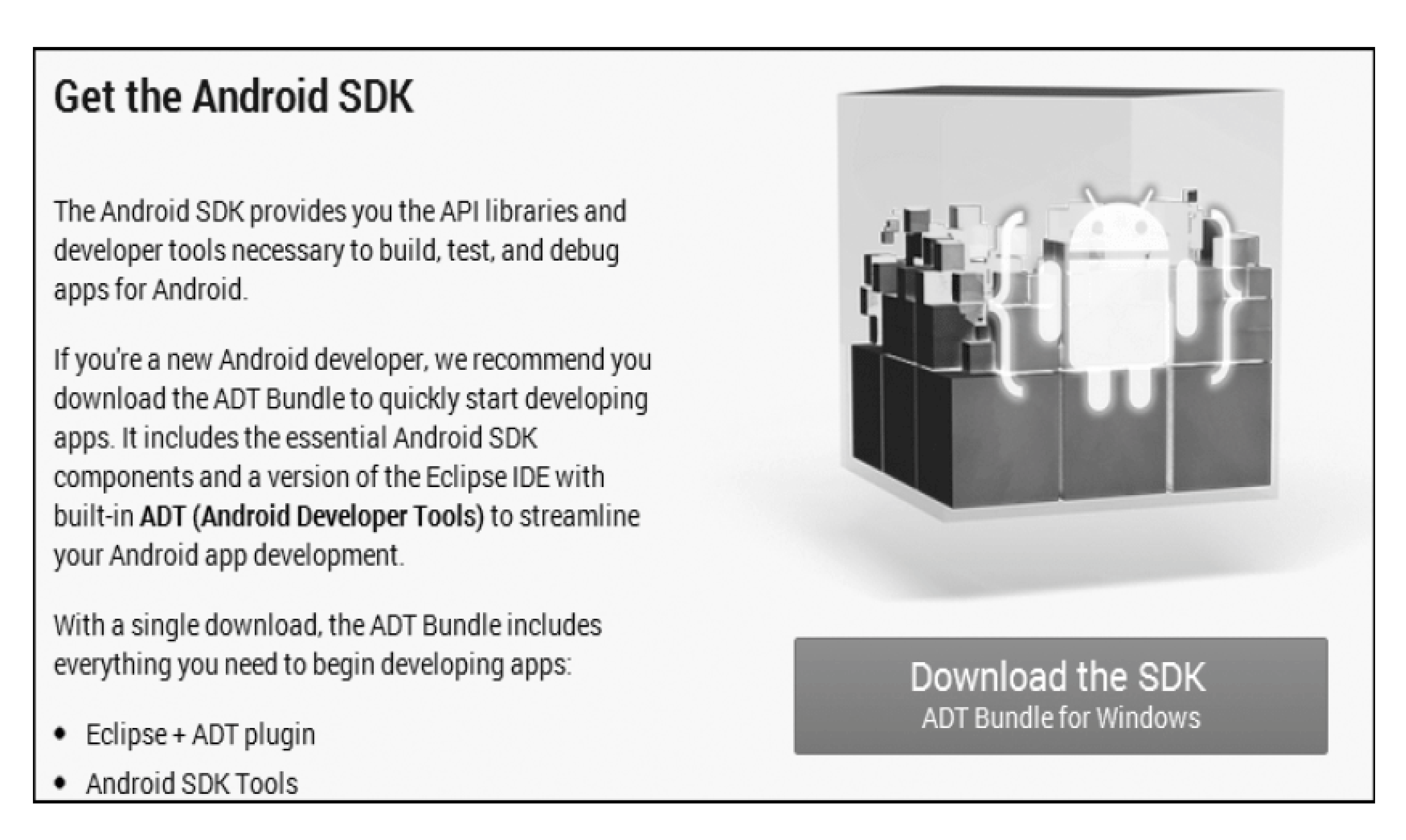


图 2.3 Android SDK 下载界面

2.1.5 Android插件 ADT

ADT 是 Eclipse、MyEclipse 等集成开发环境的 Android 开发插件，可以在安装 Eclipse、MyEclipse 之后加载插件，不需要单独下载。

2.2 Windows+Eclipse 开发环境配置过程

下面介绍 Windows 环境下如何在 Eclipse 开发环境中配置 Android。

2.2.1 安装 JDK

JDK 的安装方法简单，本书中不做详细说明。安装完成后，添加以下环境变量。

(1) Java_HOME：D:\Program Files\Java\jdk1.7.0_15(安装 JDK 的目录)。

(2) CLASSPATH：.;%Java_HOME%\lib\。

(3) Path：在开始追加 %Java_HOME%\bin。

注：设置环境变量对搭建 Android 开发环境不是必需的，可以跳过。

安装完成之后，可以检查 JDK 是否安装成功。打开 cmd 窗口，输入"java -version"查看 JDK 的版本信息。出现类似图 2.4，表示安装成功。

```
命令提示符
Microsoft Windows [版本 6.1.7600]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\xulin>java -version
java version "1.7.0_15"
Java(TM) SE Runtime Environment (build 1.7.0_15-b03)
Java HotSpot(TM) Client VM (build 23.7-b01, mixed mode, sharing)

C:\Users\xulin>_
```

图 2.4 验证 JDK 安装是否成功

2.2.2　安装 Android SDK

解压缩安装包,得到一个包含 eclipse、sdk 和 SDK Manager.exe 的文件夹。文件夹如图 2.5 所示。

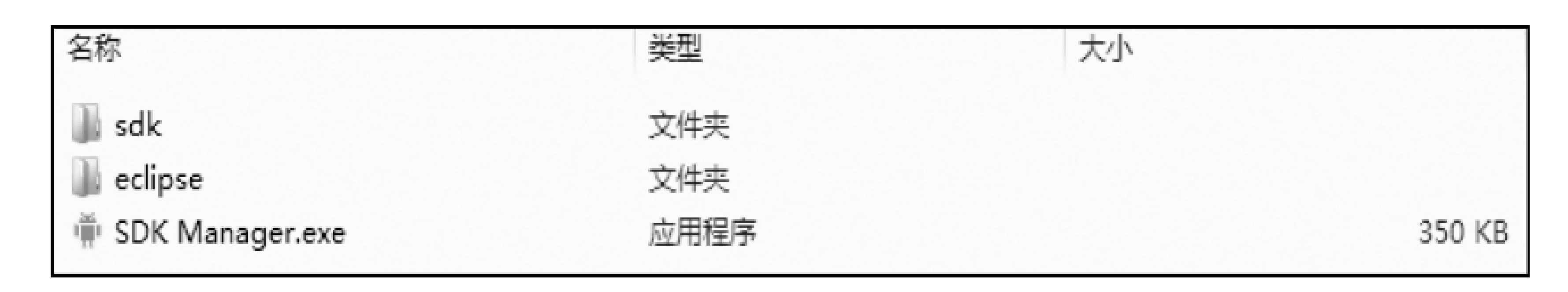


图 2.5　Android SDK 安装文件夹

由文件夹可以看到,使用新版本 SDK 不需要下载 Eclipse,而 Eclipse 中 Android ADT 已经加载,不需要像老的版本一样手动加载 Android ADT。所以,使用最新版本的 SDK,在安装好 SDK 之后,就可以开始 Android 开发了。

运行 SDK Manager.exe,出现对话框如图 2.6 所示的。

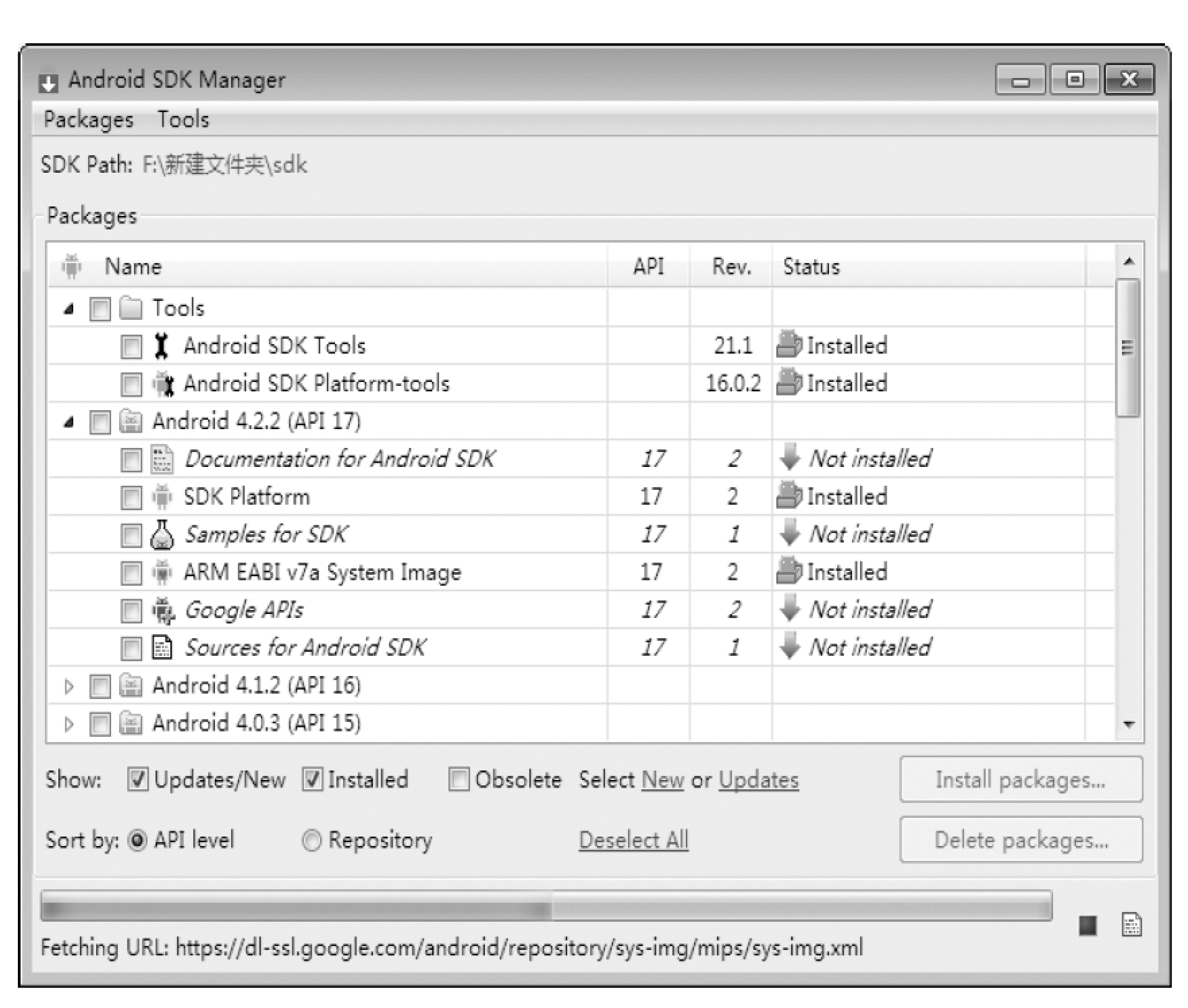


图 2.6　Android SDK Manager

选中所需要的 Packages(Android 版本),这里选择了全部的包,单击 Install 59 packages(59 是选择的包的个数),出现如图 2.7 所示对话框。

单击 Android Googletv License 或者 Android SDK License,选择 Accept License,即可选择 Android Googletv License 或者 Android SDK License 的全部。当选择 Android Googletv License 或者 Android SDK License 中的某一项时,单击 Accept 单选按钮即可选择此项,单击 Reject 单选按钮即可取消选择此项,如图 2.8 所示。

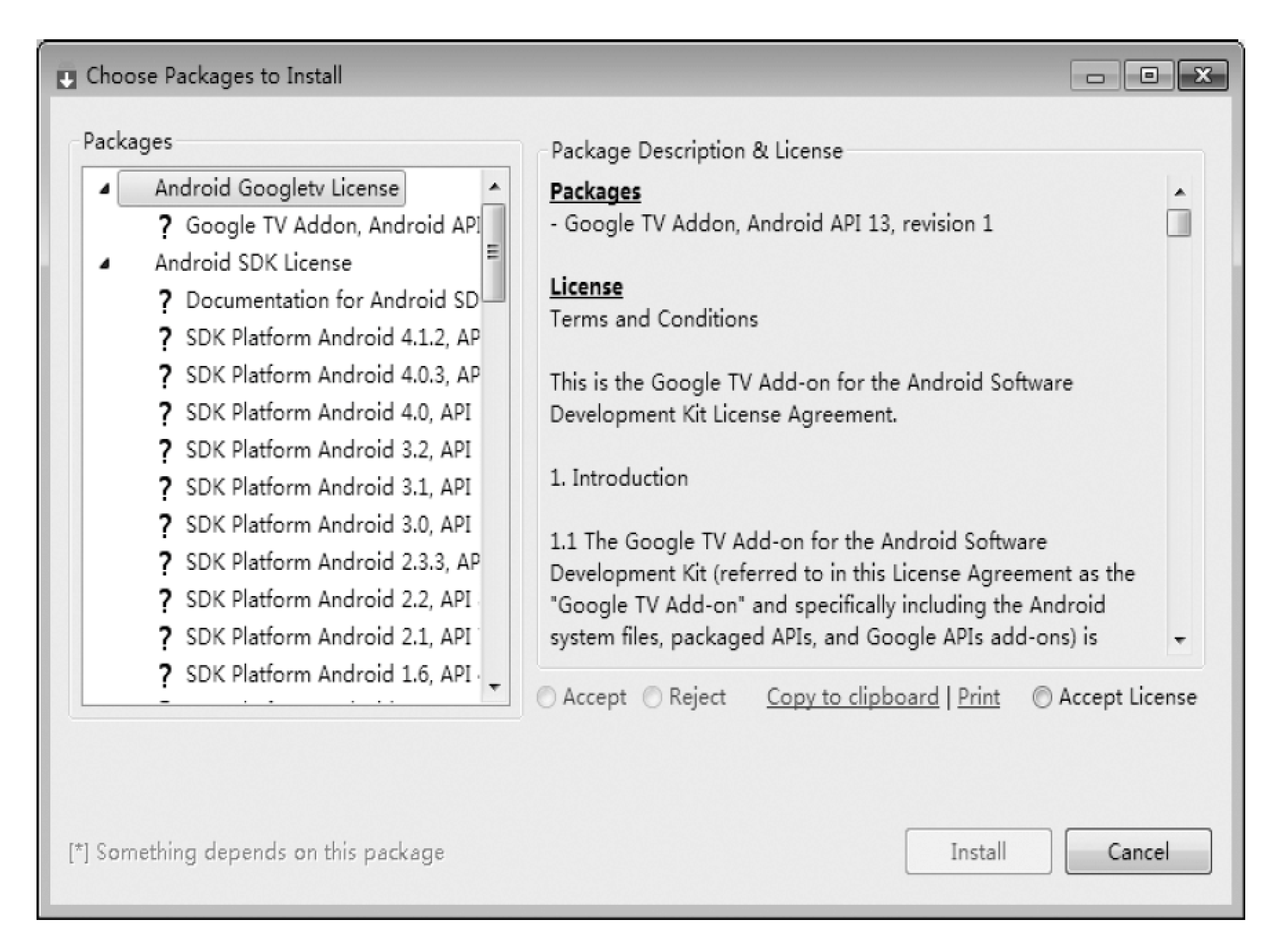


图 2.7　Choose Packages to Install(1)

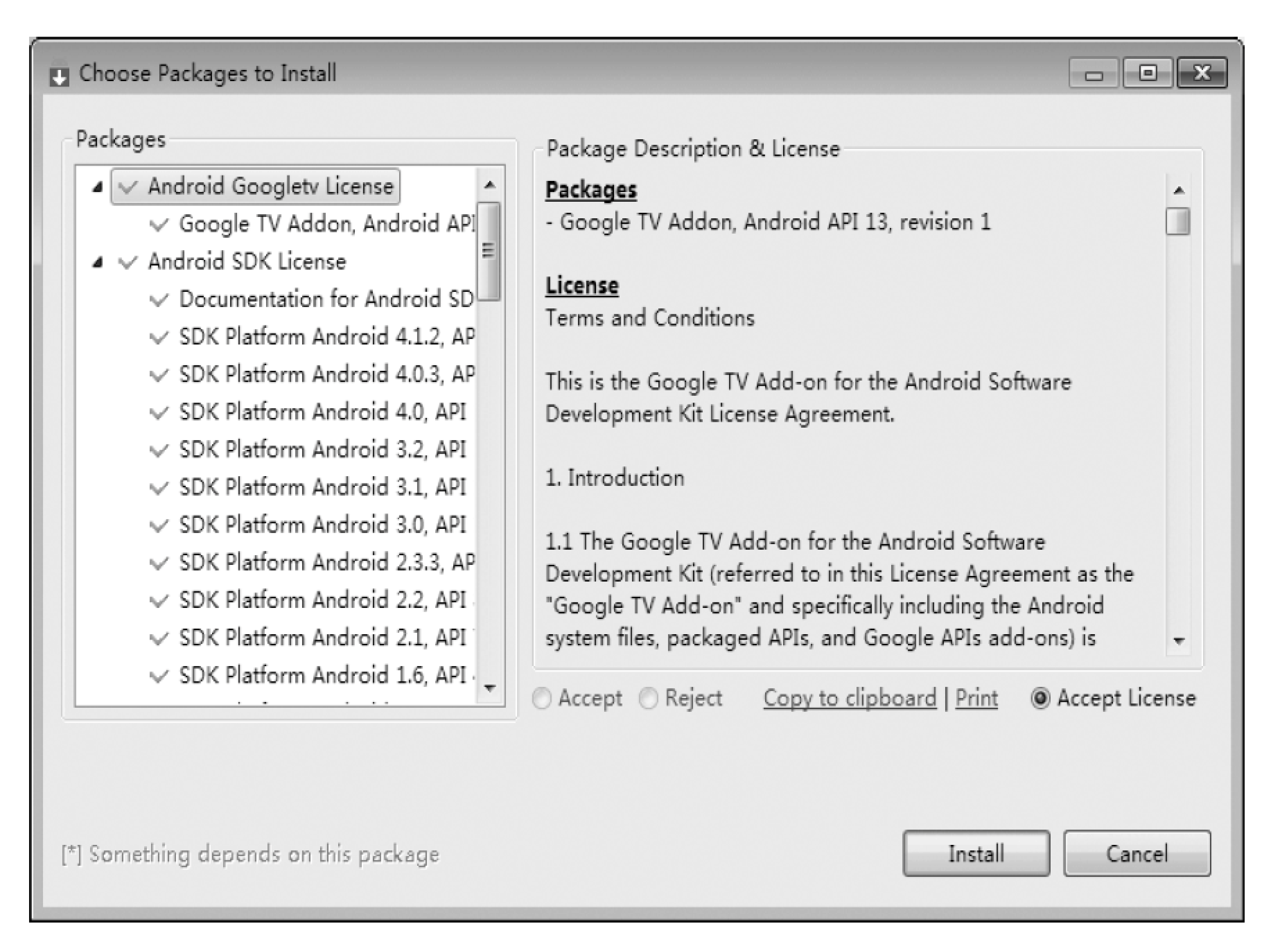


图 2.8　Choose Packages to Install(2)

单击 Install 按钮,SDK 开始安装,出现如图 2.9 所示的对话框。

安装完成后,在用户变量中新建 PATH 值为：Android SDK 中的 tools 绝对路径(本机为 F:\eclipse\sdk\tools)。

重启计算机以后,进入 cmd 命令窗口,检查 SDK 是否安装成功。运行 android -h,出现类似图 2.10 所示的画面,表示安装成功。

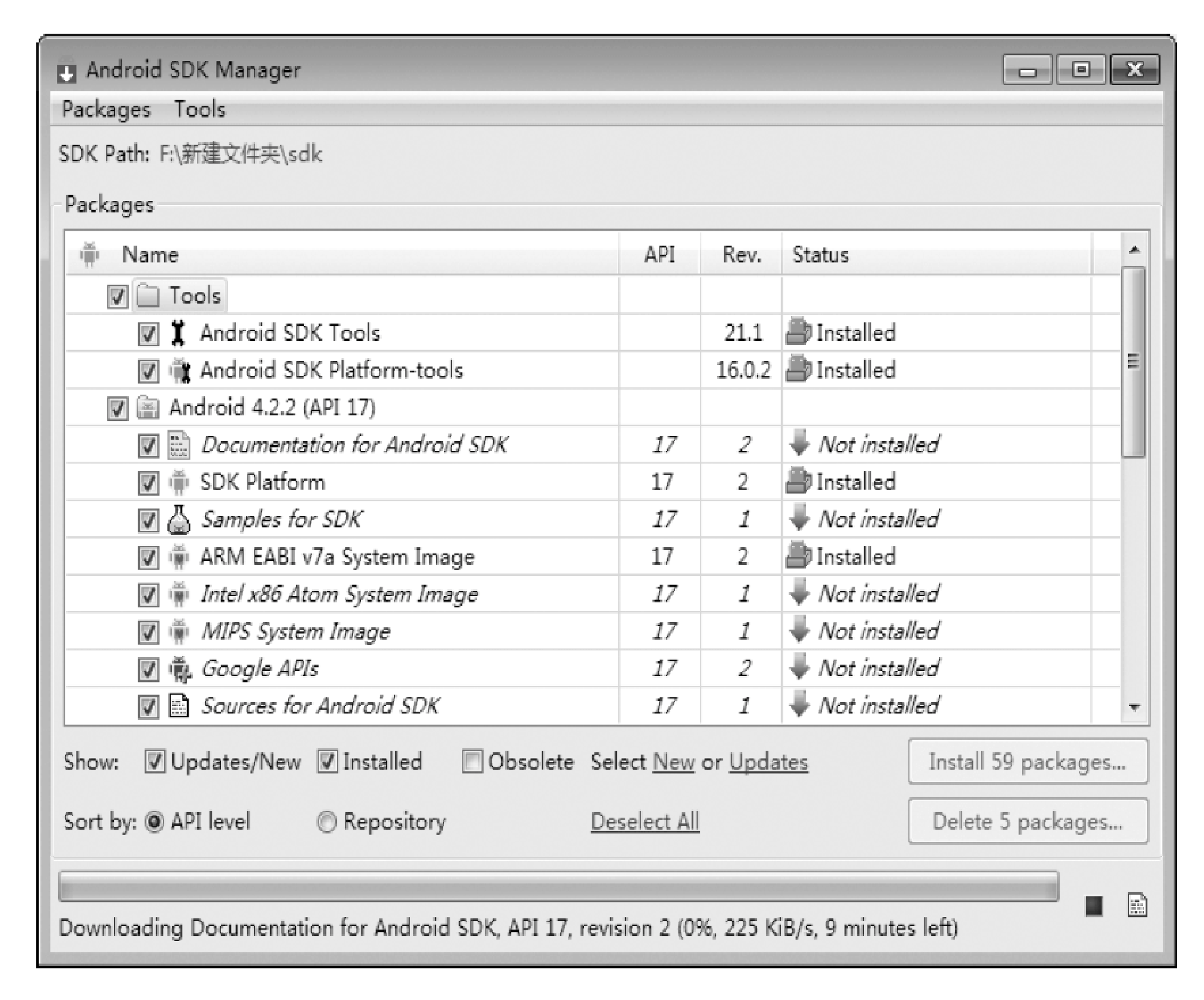


图 2.9 Android SDK Manager

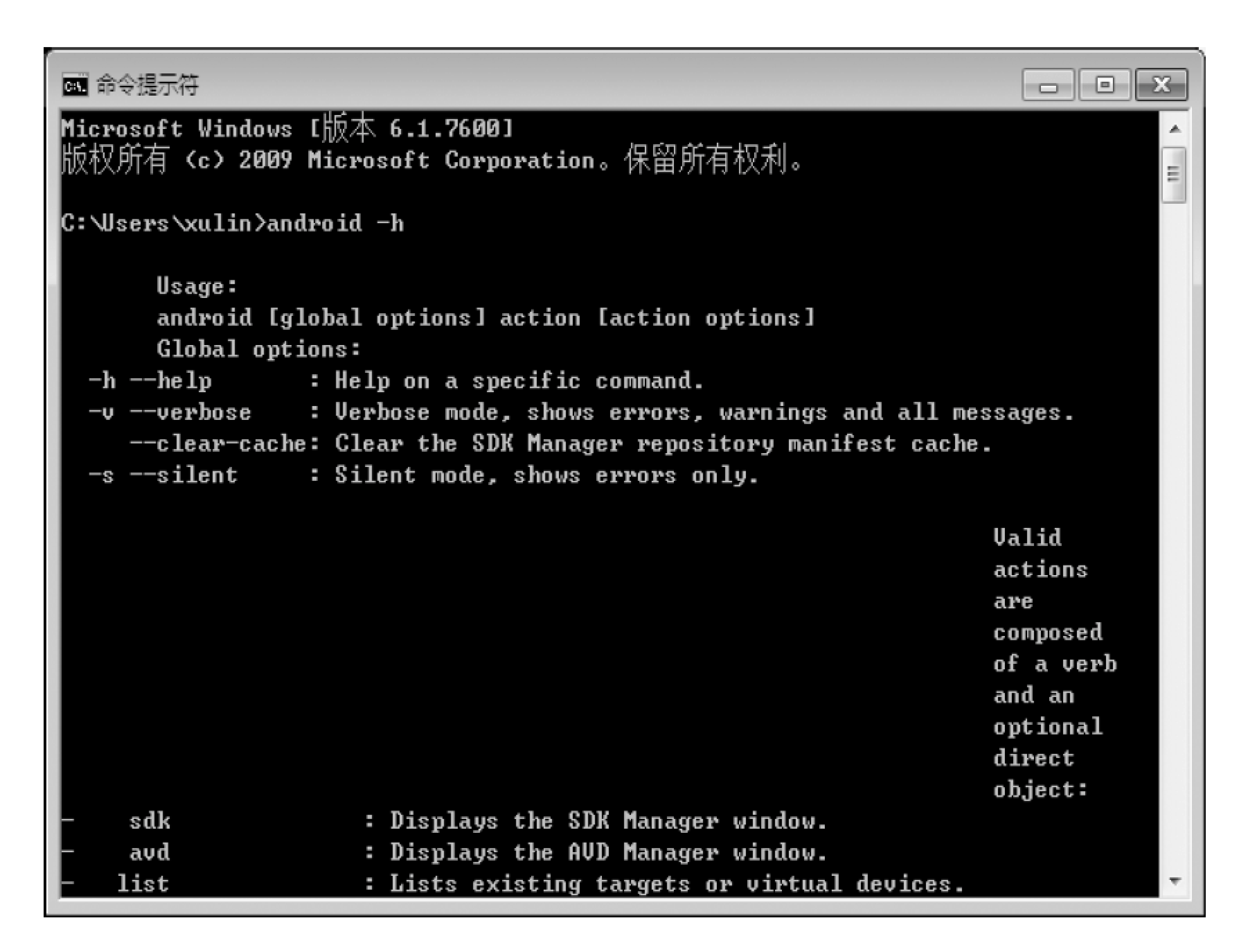


图 2.10 验证 Android SDK 是否安装成功

安装完成后，Android-sdk-windows 文件夹如图 2.11 所示。

如图 2.12 所示，打开解压目录下的 eclipse 目录下的 eclipse.exe，进入 Eclipse 开发环境。单击菜单 Window→Preferences，可以看到 SDK Location 已经设置好了。

名称	类型	大小
add-ons	文件夹	
docs	文件夹	
extras	文件夹	
platforms	文件夹	
platform-tools	文件夹	
samples	文件夹	
sources	文件夹	
system-images	文件夹	
temp	文件夹	
tools	文件夹	

图 2.11　Android SDK 文件夹

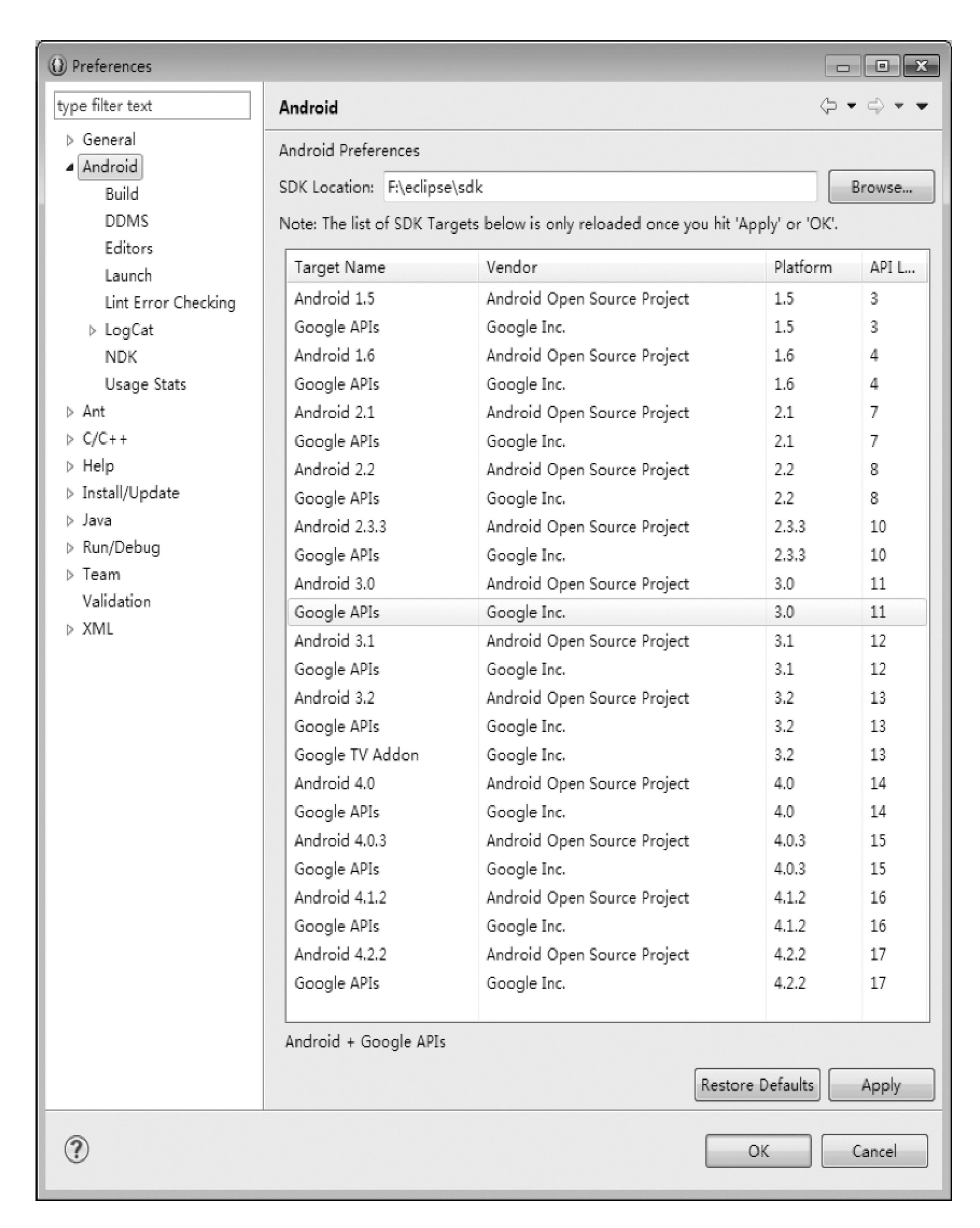


图 2.12　Preferences 对话框

223 升级 SDK 版本

选择菜单 Window→Android SDK Manager,出现如图 2.13 所示的对话框。

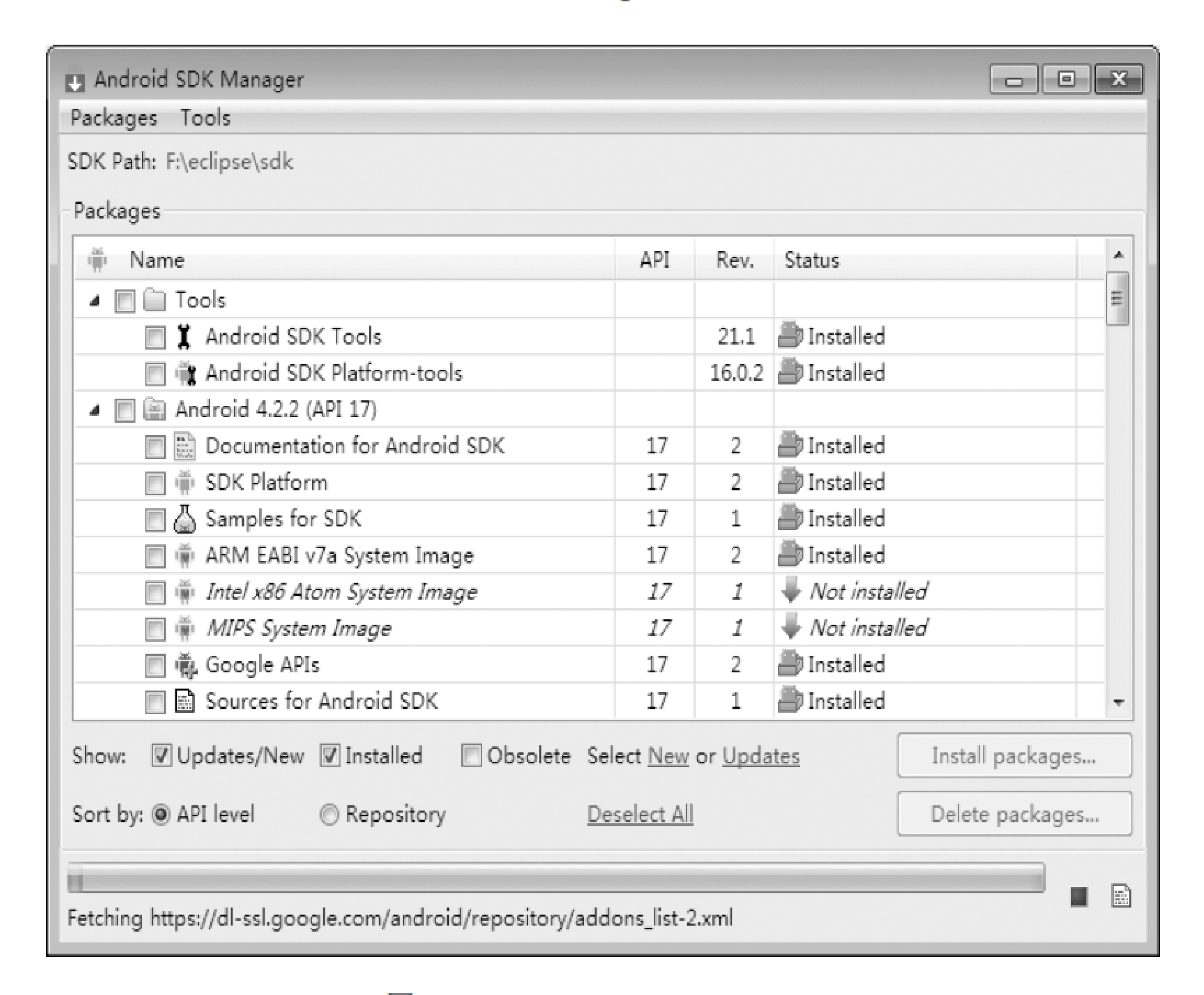


图 2.13 Android SDK Manager

操作步骤与安装 SDK 时的操作步骤相同,不再赘述。

224 新建 AVD

AVD 的全称为 Android Virtual Device,是 Android 运行的虚拟设备,是 Android 的模拟器识别。建立的 Android 要运行,必须创建 AVD,每个 AVD 上可以配置很多的运行项目。创建 AVD 的方法有两种:一是通过 Eclipse 开发环境;二是通过命令行。

1. 命令行创建

在命令行方式中找到 Tools 的路径,输入命令“Android create avd --target 2 --name my_avd”,其中,Android 是命令,后面是参数;create avd 是创建 AVD;target 2 是等级;name 是 Avd 的名称。

2. Eclipse 开发环境

选择菜单 Window→Android Virtual Device Manager,进入如图 2.14 所示的对话框。

单击 New 按钮,进入如图 2.15 所示的对话框。

AVD Name(AVD 名称)可以随便取;Device 选择手机屏幕类型;Target 选择需要的 SDK 版本;SD 卡大小自定义,设置完成后的界面如图 2.16 所示。

单击 OK 按钮,即可创建完成,完成后的界面如图 2.17 所示。

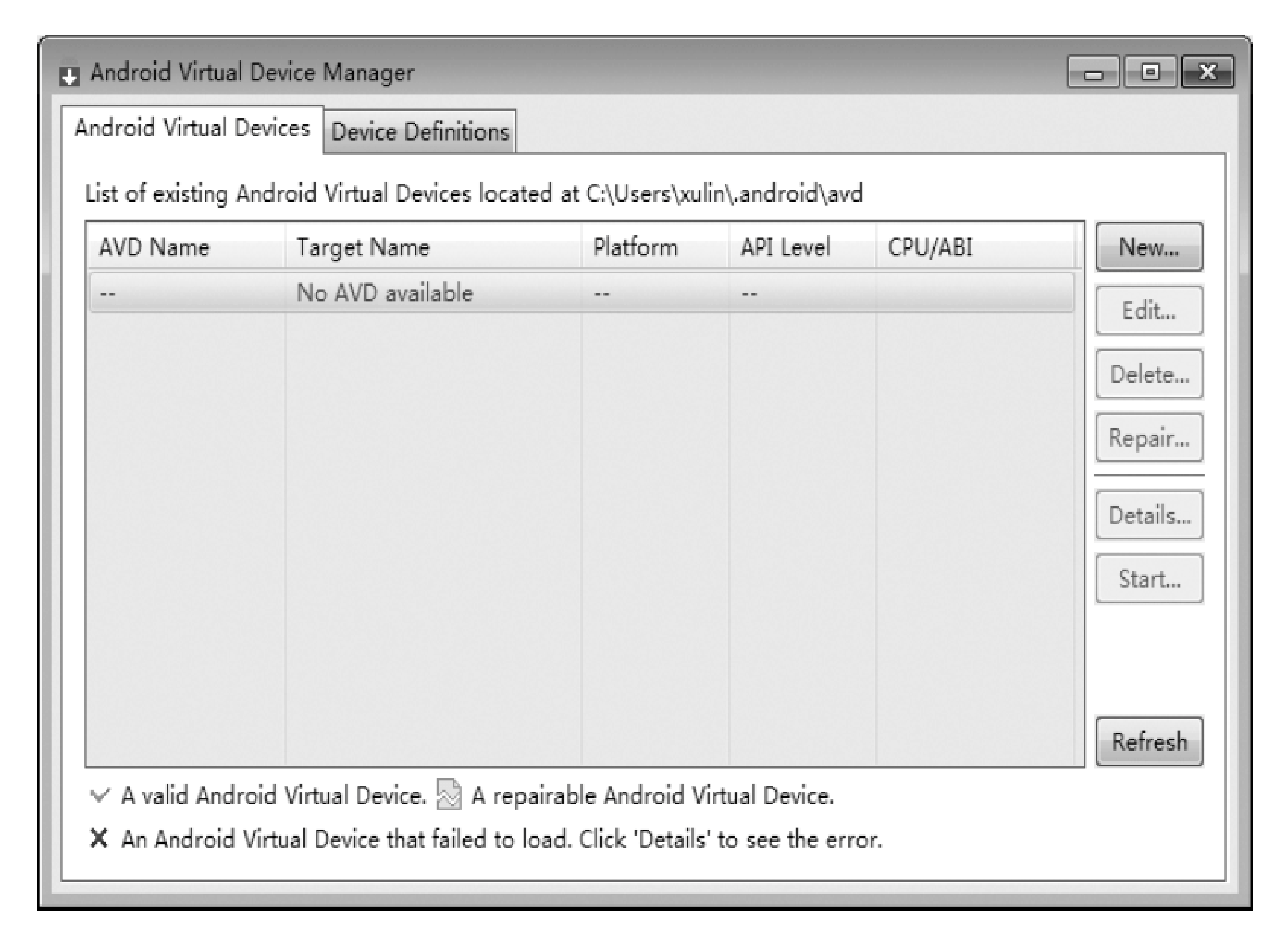


图 2.14 Android Virtual Device Manager

图 2.15 Create new Android Virtual Device (AVD)

图 2.16 Edit Android Virtual Device (AVD)

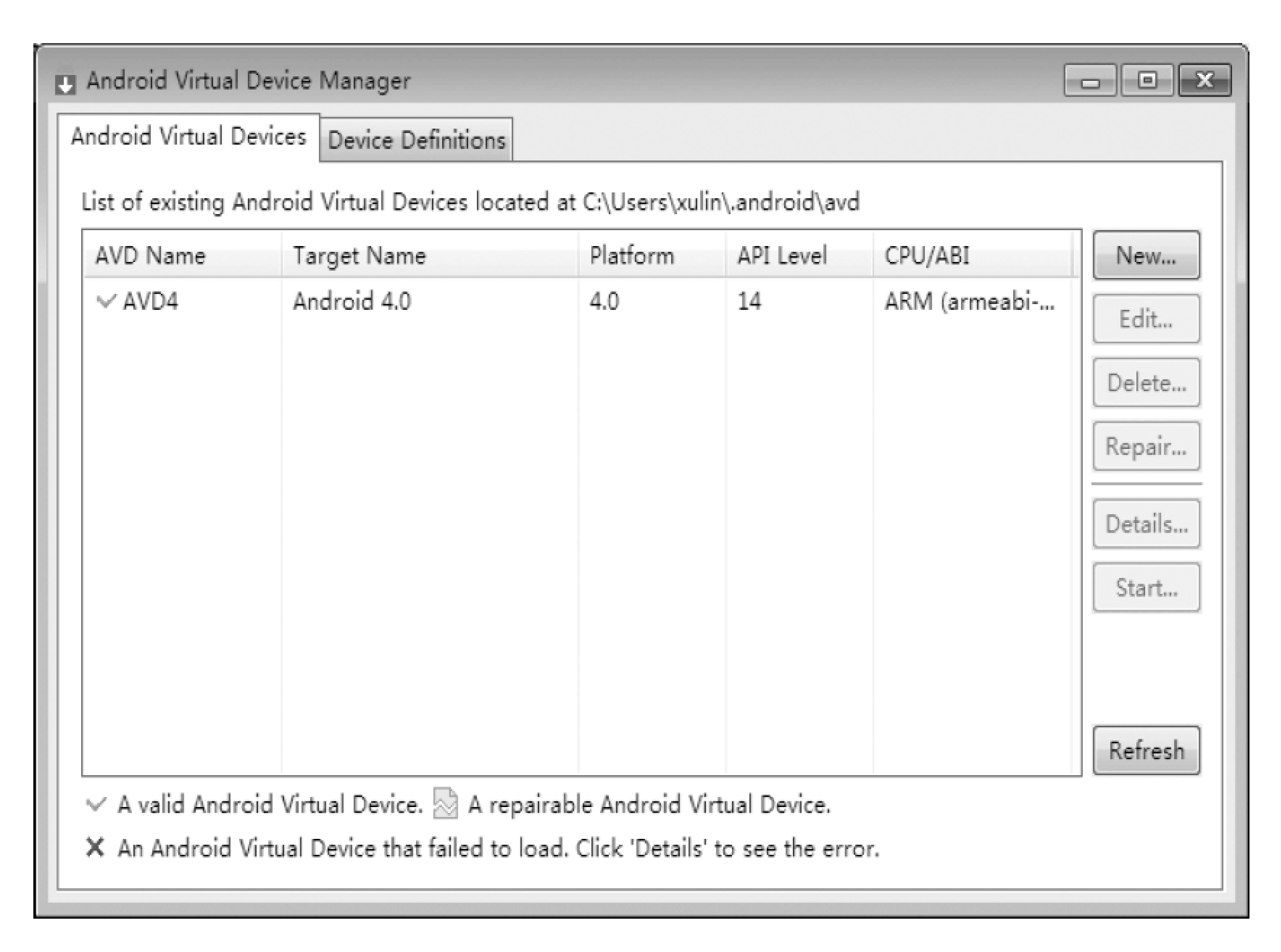


图 2.17　Android Virtual Device Manager

2.3　Windows+MyEclipse 开发环境配置过程

MyEclipse 的安装和配置相对复杂。JDK、Android SDK 的安装和 Eclipse 配置一样。MyEclipse 的安装只需要按照提示一步步操作即可。安装完成 MyEclipse 之后，需要手动安装 Android ADT。安装过程分为两种：一种是在线；另一种是离线。在线的耗费时间非常多。离线的安装比较简单。下面首先介绍离线安装。

2.3.1　Android ADT 离线安装

Android ADT 的下载地址是 http://developer.Android.com/sdk/installing/installing-adt.html。

本书下载的版本是 ADT-21.1.0.zip。解压缩后目录如图 2.18 所示。

名称	修改日期	类型	大小
features	2013/2/6 0:46	文件夹	
plugins	2013/2/6 0:46	文件夹	
web	2013/2/6 0:46	文件夹	
index	2013/2/6 0:46	Chrome HTML D...	2 KB
site	2013/2/6 0:46	XML 文档	2 KB

图 2.18　Android ADT 文件夹

将其中的 features、plugins、web 3 个文件夹复制到 Myeclipse/dropins 目录下。重启 MyEclipse。进入 MyEclipse 后会弹出如图 2.19 所示的对话框。

单击 Open Preferences 按钮，进入如图 2.20 所示对话框。

单击 Proceed 按钮，进入如图 2.21 所示的对话框。

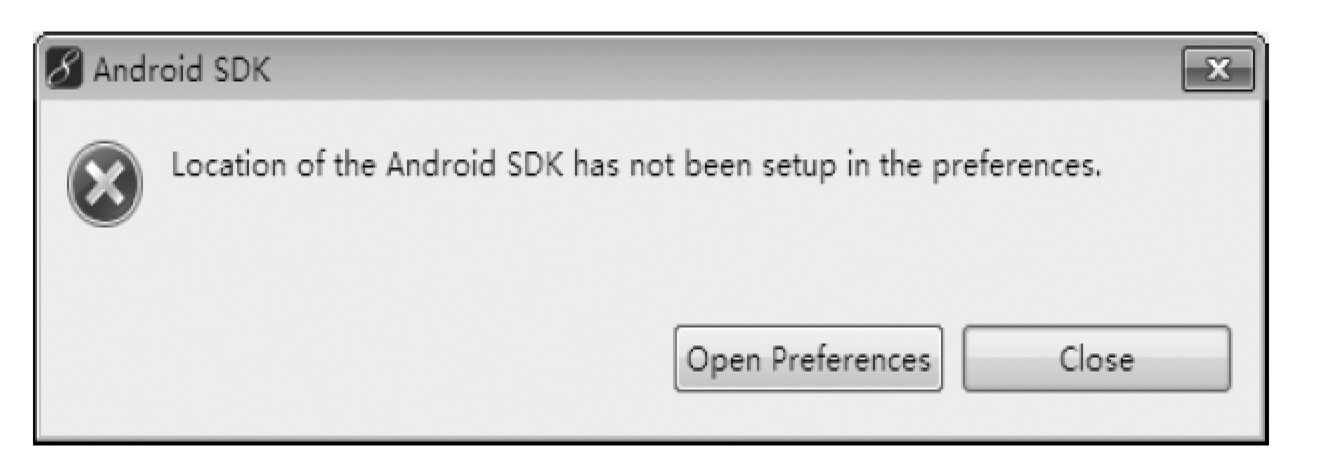


图 2.19　MyEclipse 提示(1)

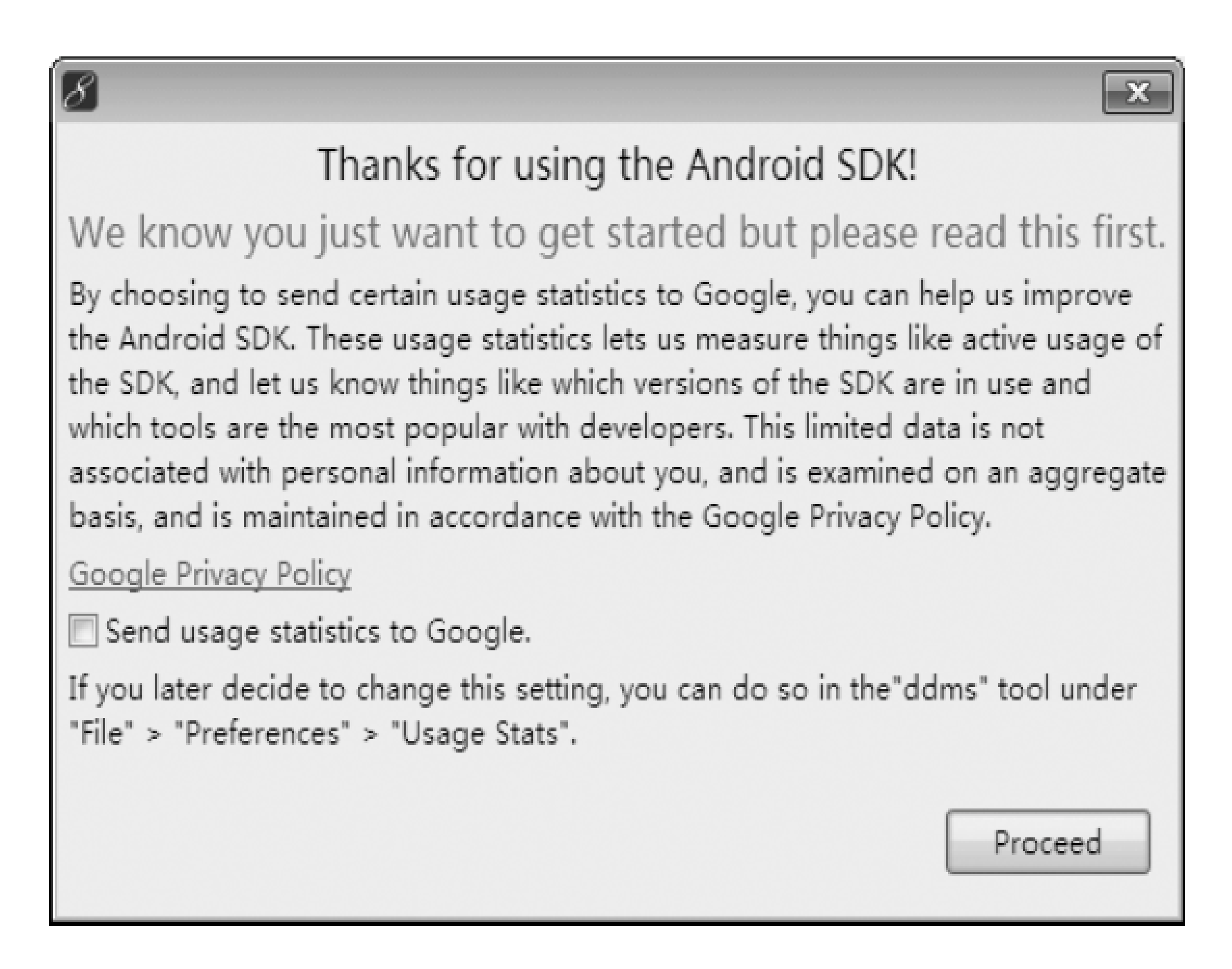


图 2.20　MyEclipse 提示(2)

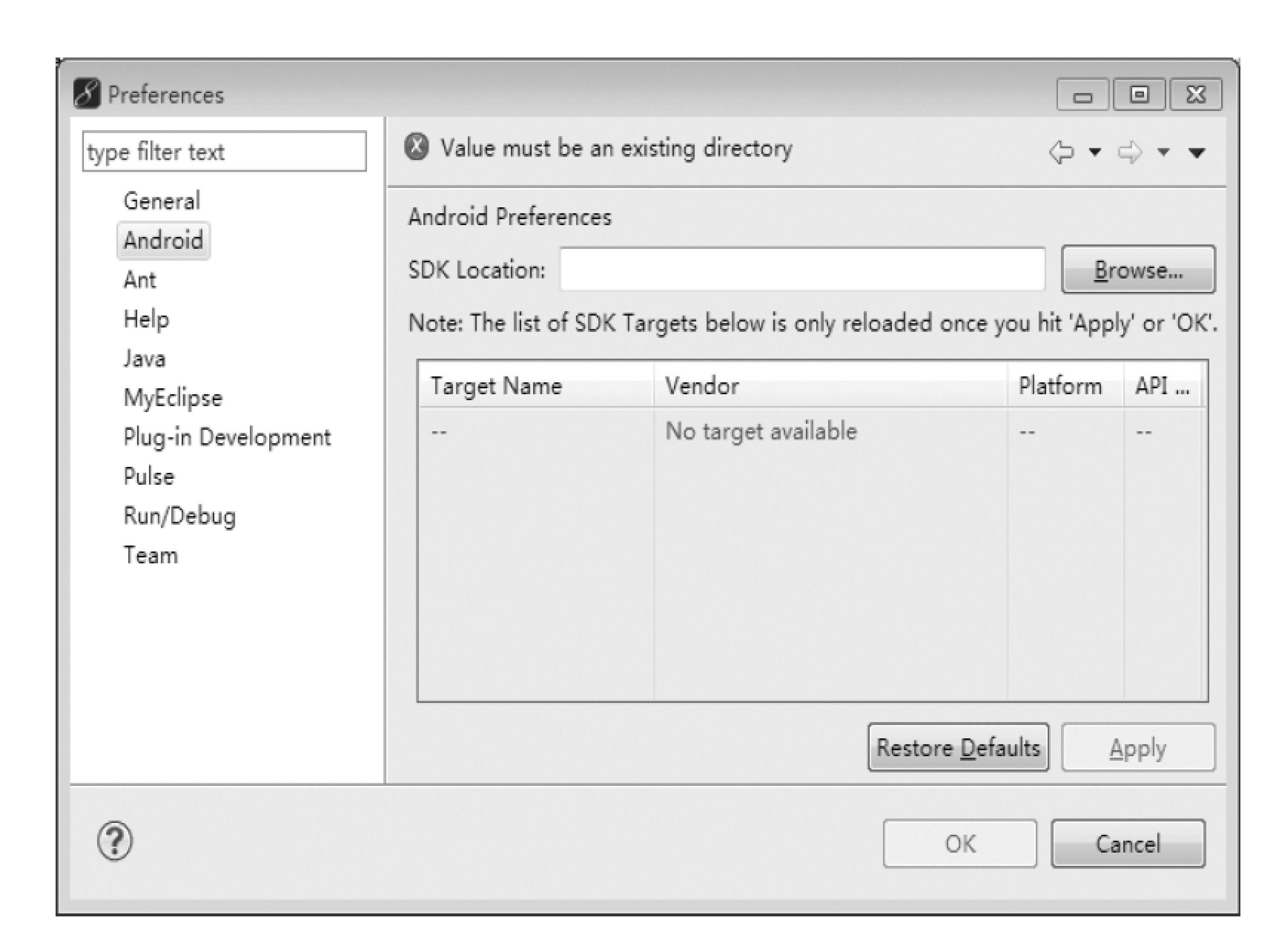


图 2.21　Preferences

在此处设置 SDK 路径，然后单击 Apply 按钮，出现如图 2.22 所示的对话框。

2.3.2　Android ADT 在线安装

打开 MyEclipse 开发环境，单击菜单 Help→MyEclipse configuration Center，打开如图 2.23 所示的界面。

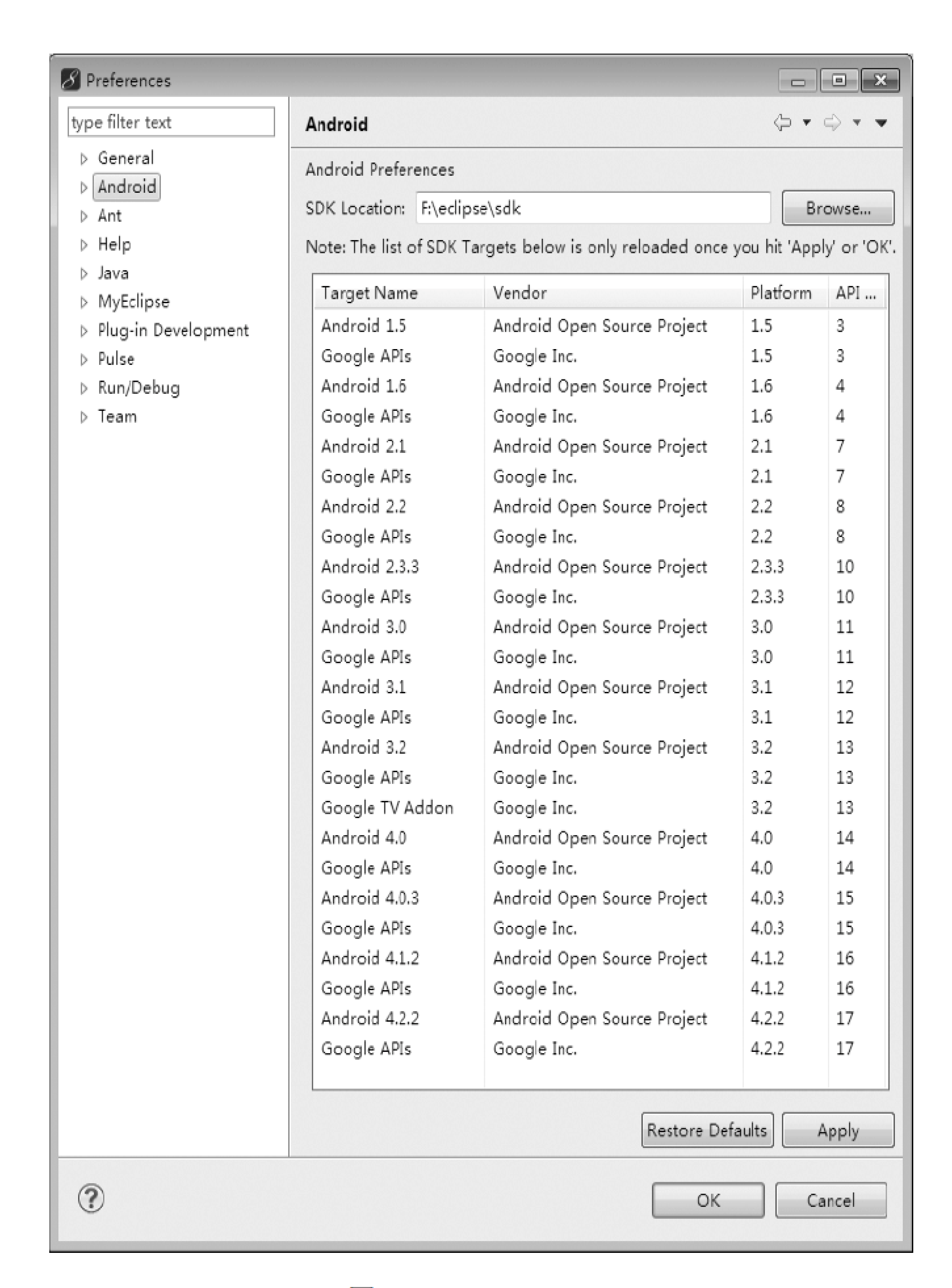


图 2.22　Preferences

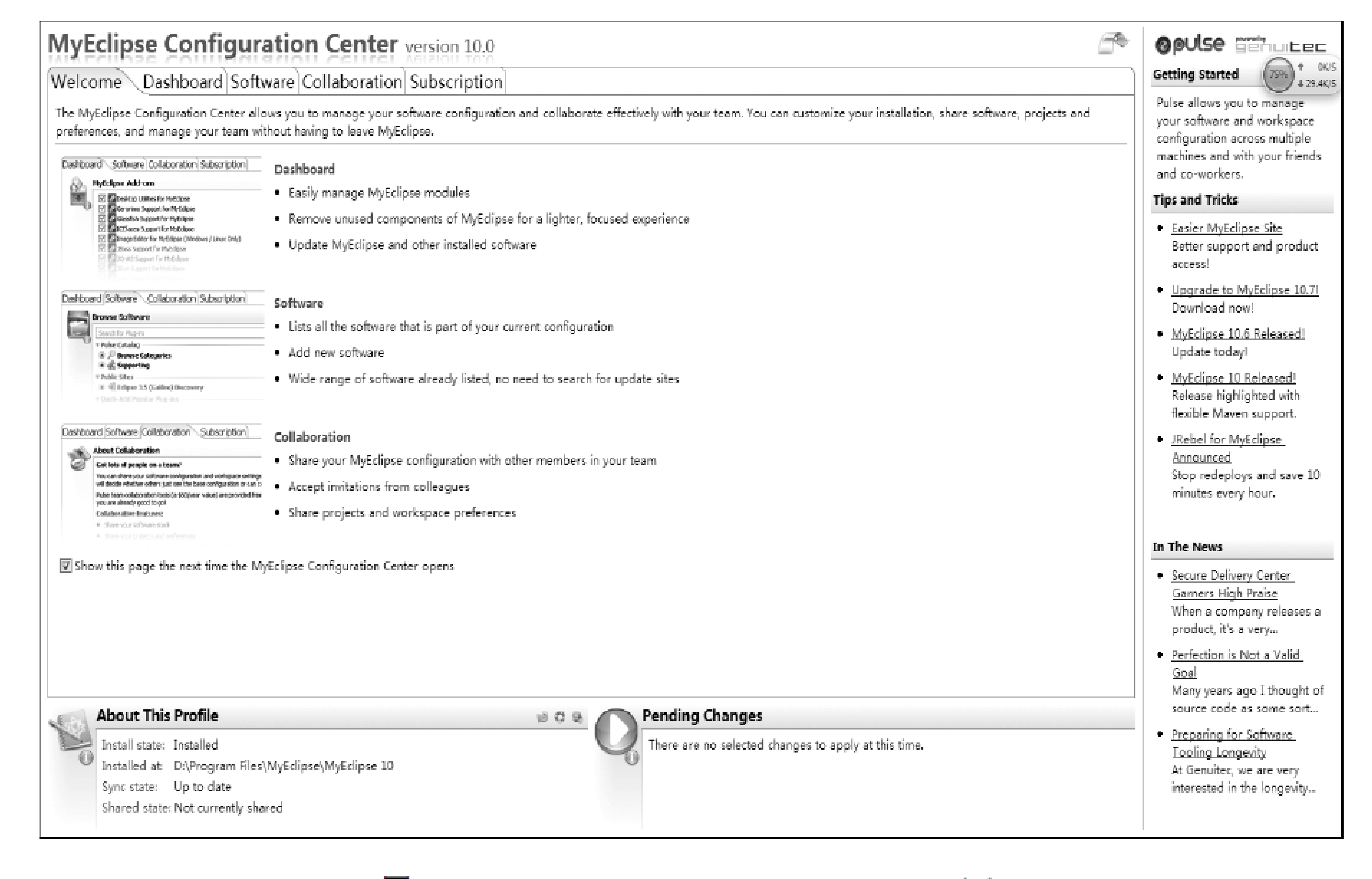


图 2.23　MyEclipse Configuration Center(1)

单击 Software 选项卡,界面如图 2.24 所示。

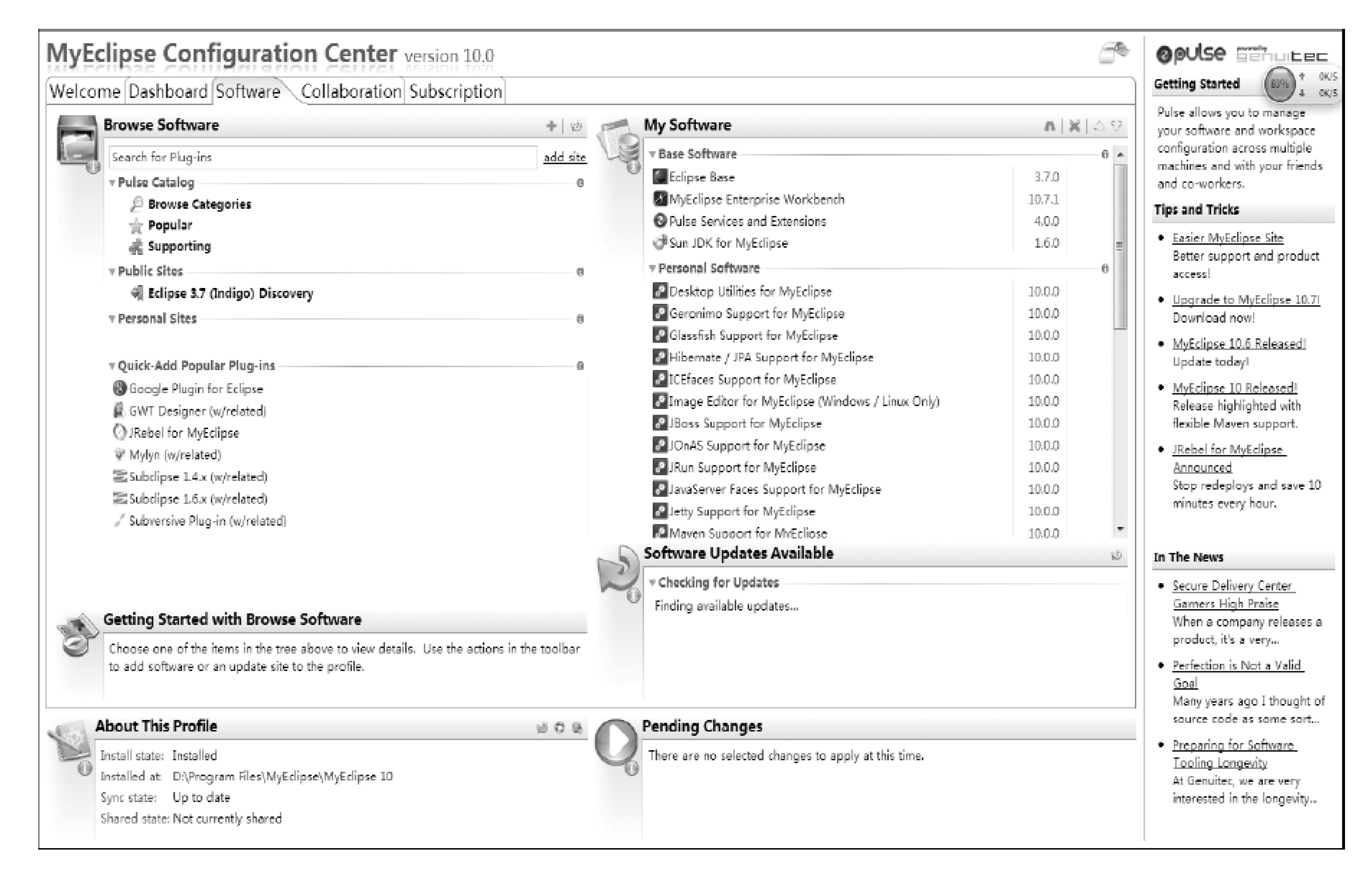


图 2.24　MyEclipse Configuration Center(2)

单击 Browse Software 后面的 add site,如图 2.25 所示。

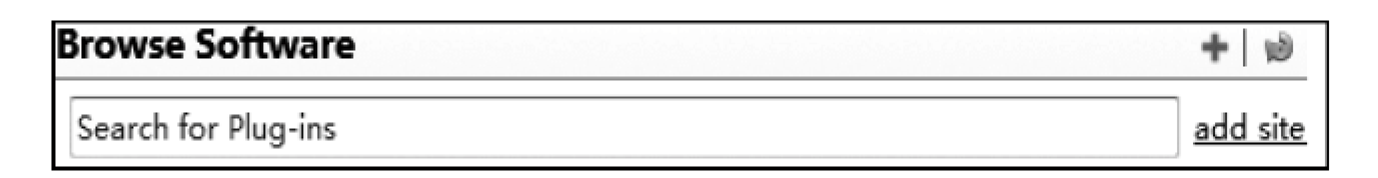


图 2.25　MyEclipse Configuration Center(3)

单击后,出现插件网址维护对话框,如图 2.26 所示。在 Name 文本框中输入“Android”(这里可以自定义)。在 URL 中输入插件网址“https://dl-ssl.google.com/android/eclipse/”(如果出错,请将 https 改成 http)。Recently Used Updates Sites 显示的是已经维护的 Site。所以,页面上在 Name 后面显示红色的×,提示已经有了相同的名字。修改 Name 后,单击 OK 按钮即可保存,如图 2.26 所示。

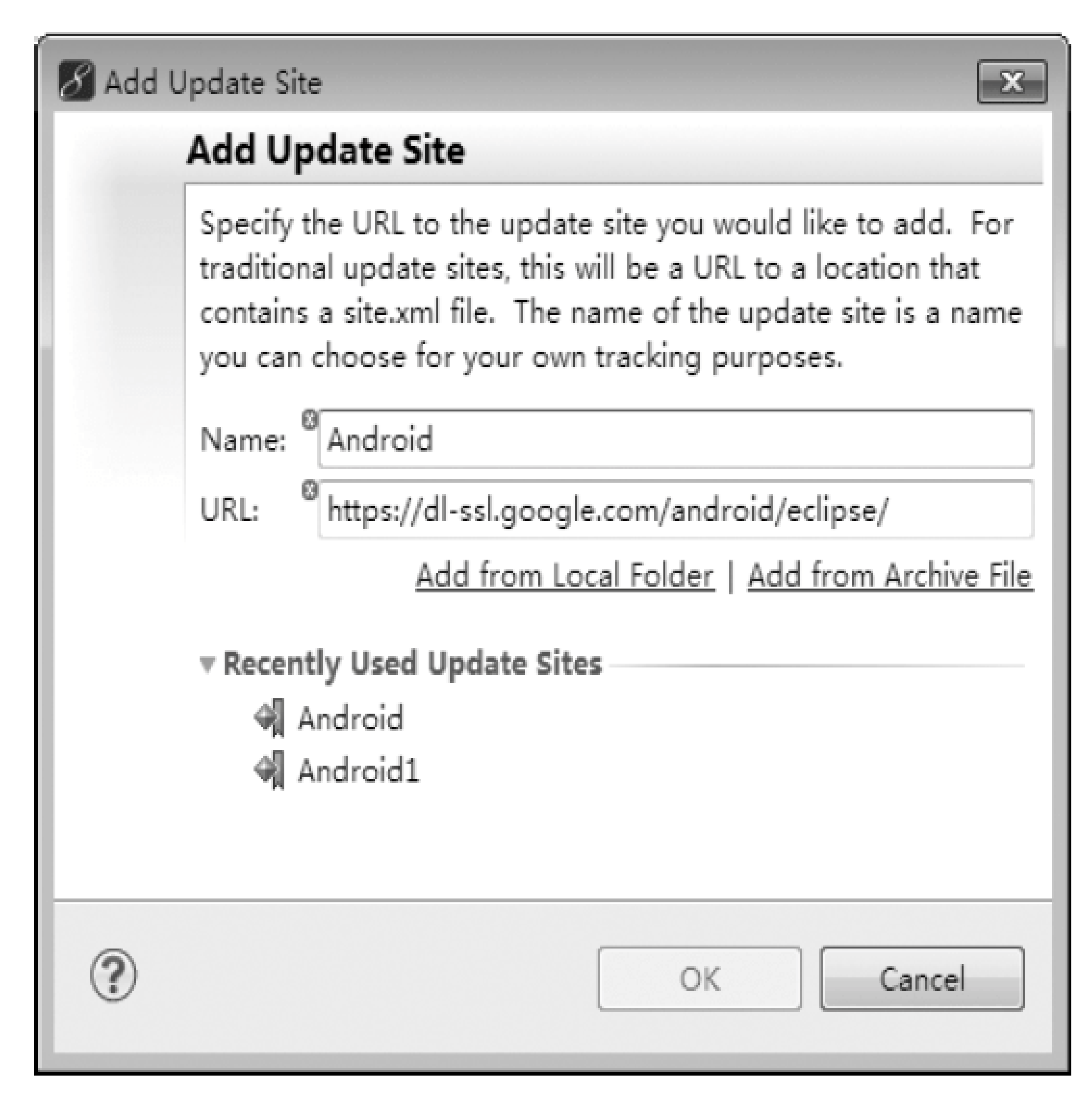


图 2.26　Add Update Site

维护 Site 后，Software 的 Browse Software 界面如图 2.27 所示。

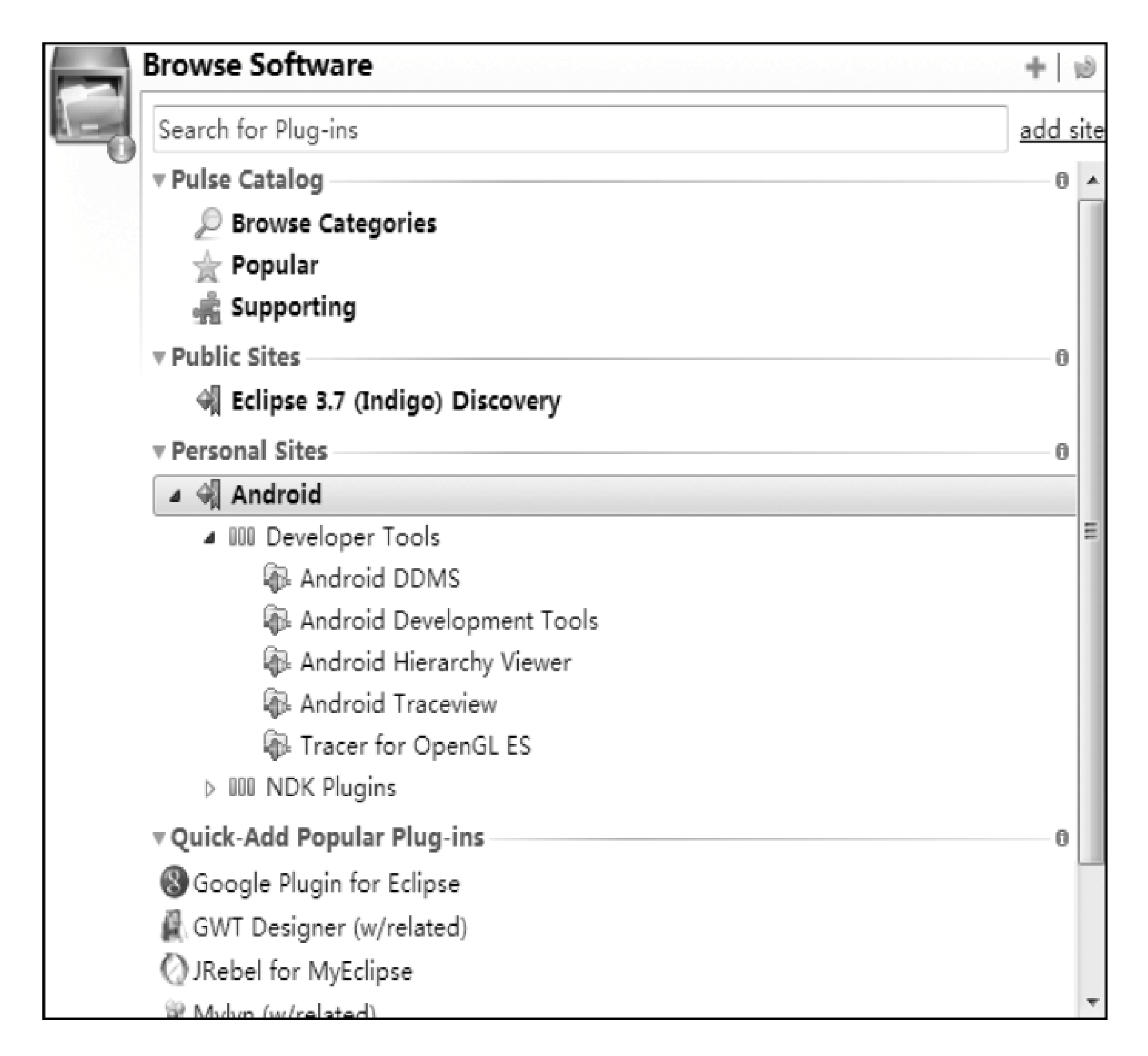


图 2.27　Browse Software(1)

在 Android 的 Developer Tools 的 Android DDMS 上右击，弹出快捷菜单，如图 2.28 所示。

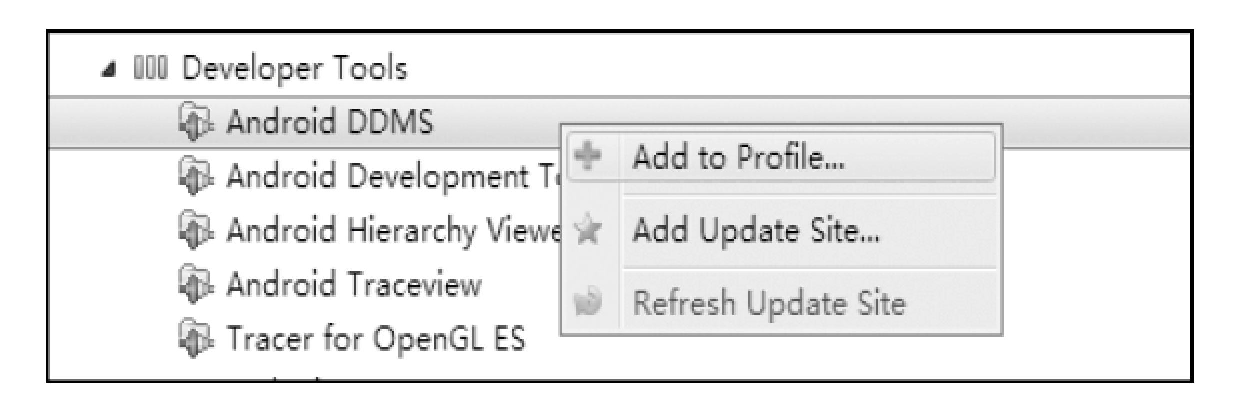


图 2.28　Browse Software(2)

选择 Add to Profile 命令，Software 的 Software Updates Available 界面变化如图 2.29 所示。

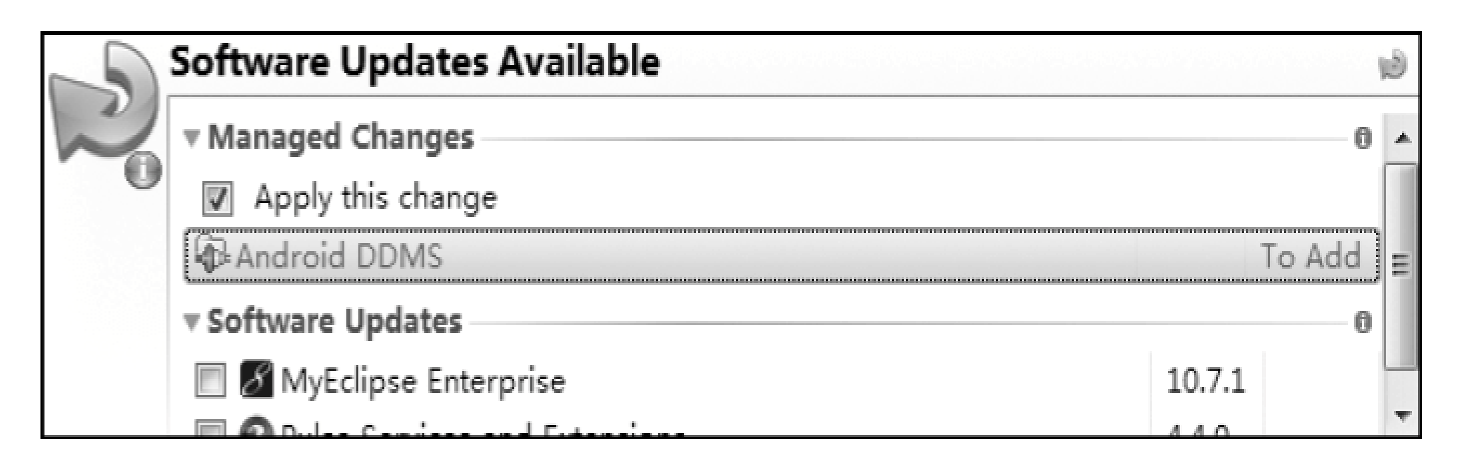


图 2.29　Software Updates Available

Software 的 Pending Changes 界面变化如图 2.30 所示。

单击 Apply 1 change 按钮后，联网下载 ADT 插件，速度比较缓慢，之后的操作按照提示操作即可。

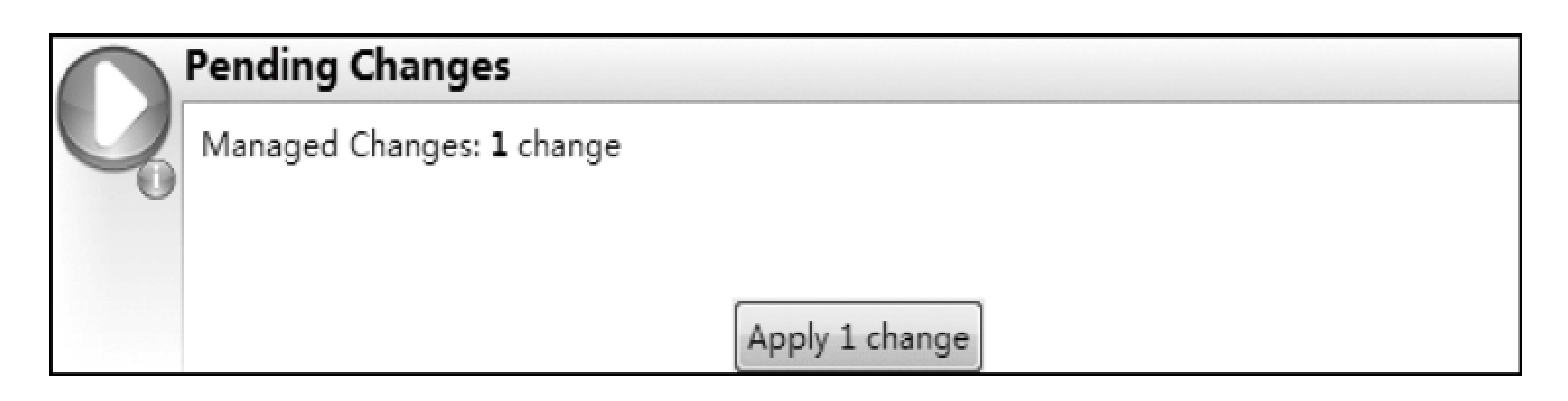


图 2.30 Pending Changes

2.4 其他开发环境配置过程

2.3 节介绍了在 Windows 下搭建 Android 的步骤，本节介绍在 Linux 系统下搭建 Android 的步骤，采用的 Linux 版本是 ubuntu-10.10-desktop-i386。在 Linux 下创建一个 Android 用户，用来管理 Android 的开发环境，把准备的软件复制到/home/Android 目录下。

2.4.1 安装 JDK

Eclipse 和 Android SDK 都依赖于 JDK 的环境，确保工作的目录是/home/Android。

安装 JDK 的操作步骤如下：

(1) 为 JDK 安装文件添加可执行权限 chmod u+x(JDK 安装文件)。

(2) 使用./(JDK 安装文件)。

(3) 配置环境变量，Linux 下需要修改/etc/profile 文件，添加 Java_HOME=安装路径。修改 Path 变量，添加 $ Java_HOME/bin。

修改后重启系统，修改生效。最后使用 Java -version 验证 JDK 是否安装成功，如图 2.31 所示。

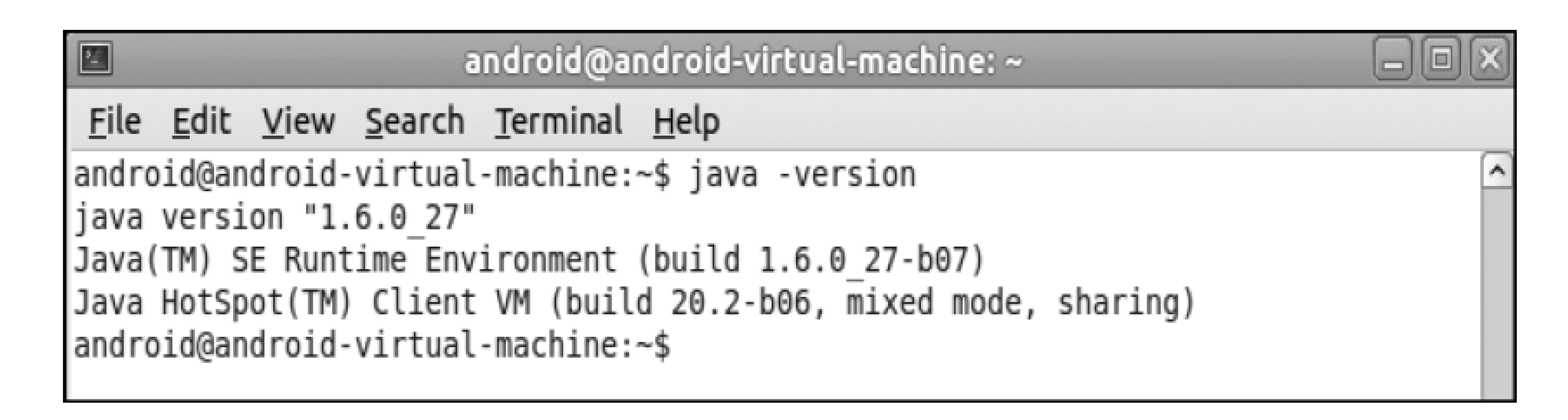


图 2.31 验证 JDK 是否安装成功

2.4.2 安装 Android SDK 并创建 AVD

安装 SDK 的操作步骤如下：

(1) 解压缩 tar -zxvf (SDK 安装文件)。

(2) 使用 Android-sdk-Linux _ x86/tools/Android 运行 Android SDK and AVD Manager，单击 Avaliable packages，选择相应的版本，单击 Install Selected 按钮进行安装，如图 2.32 所示。

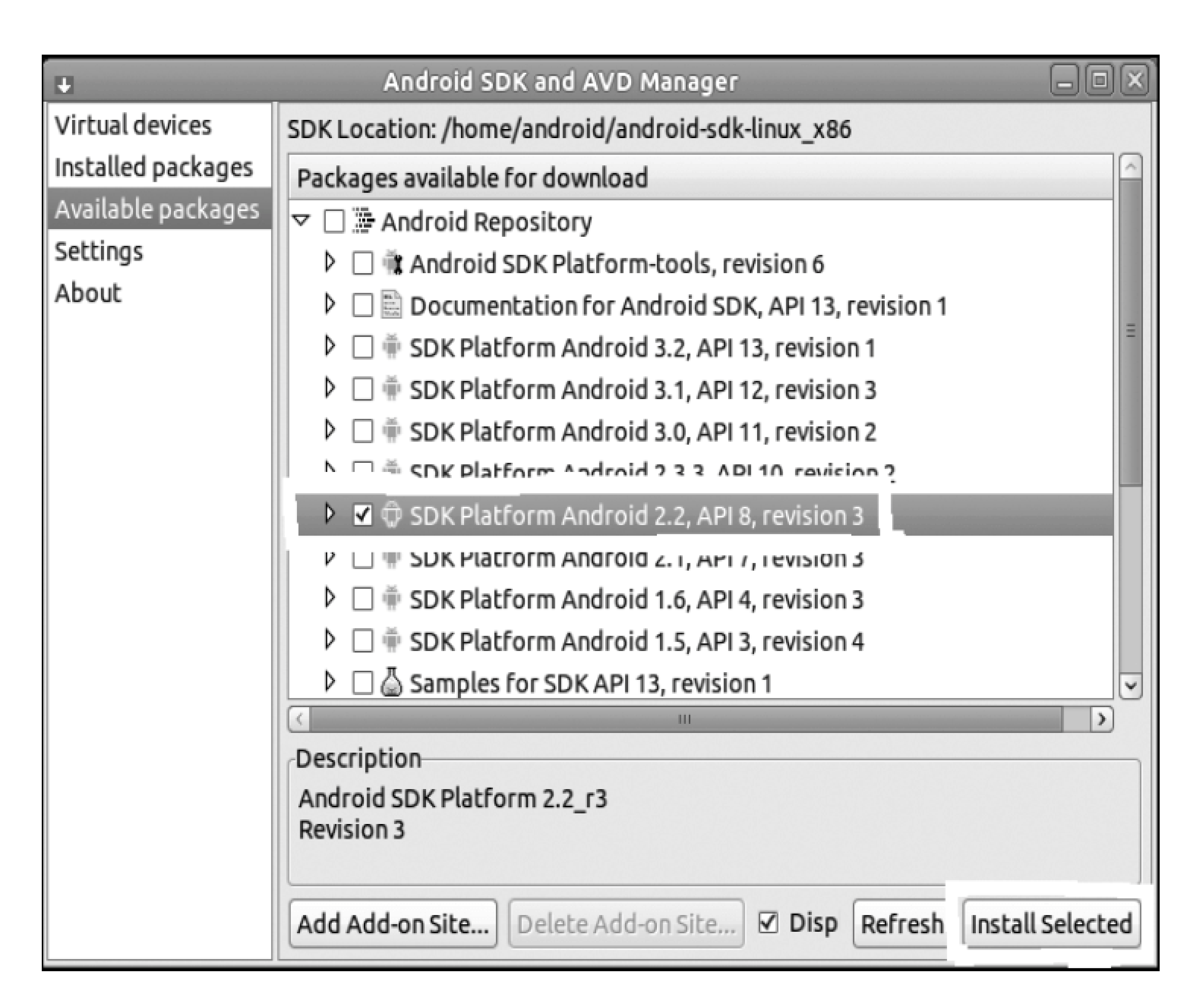


图 2.32 Android SDK and AVD Manager(1)

(3) 安装完成后单击 Virtual devices 新建一个 Android 虚拟机,如图 2.33 所示。

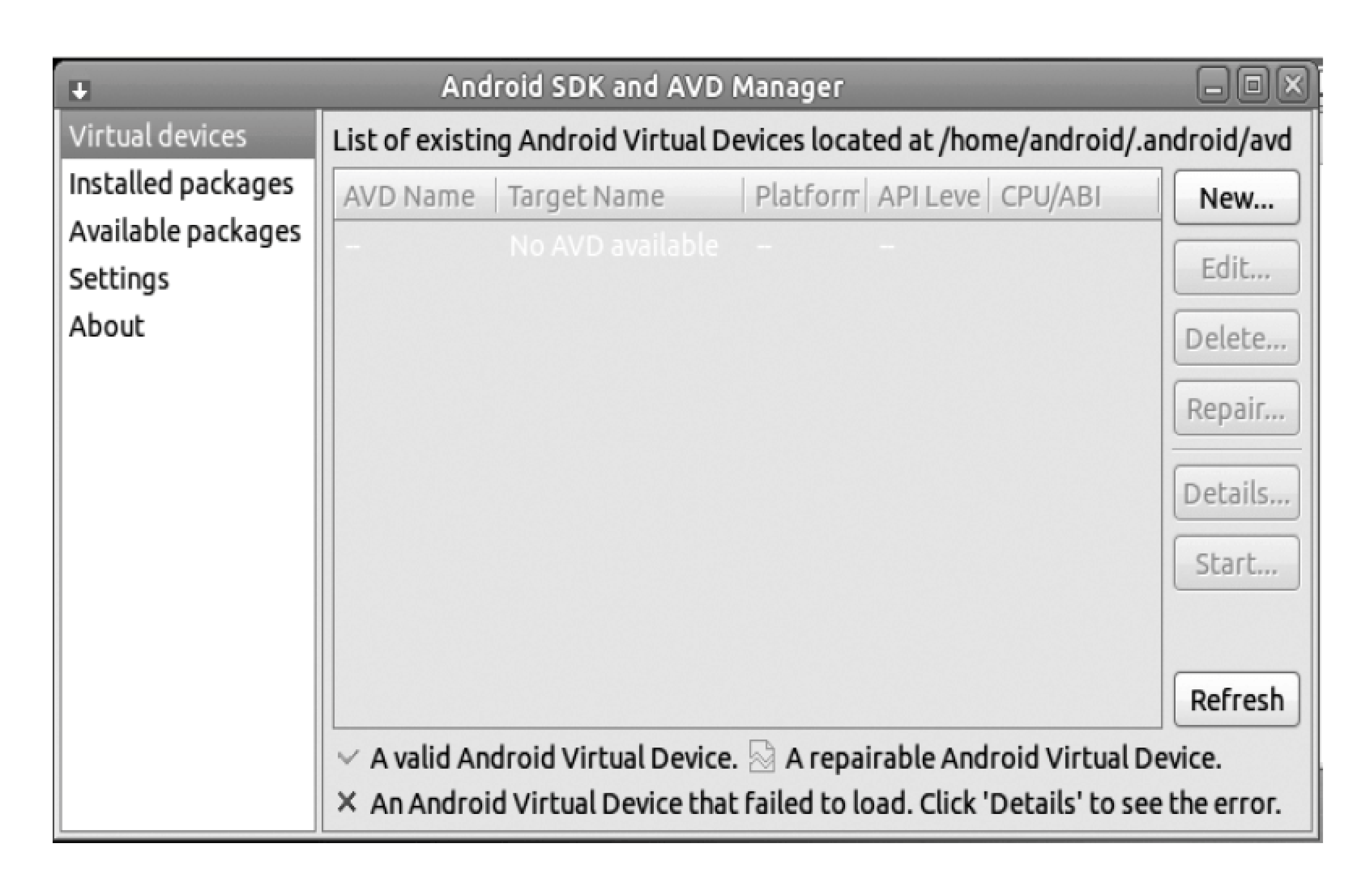


图 2.33 Android SDK and AVD Manager(2)

单击 New 按钮,打开如图 2.34 所示的对话框。

在 Name 文本框中输入 AVD 名字。在 Target 中选择要支持的 Android 版本。在 SD Card 中输入自定义的大小。

2.4.3 安装 Eclipse

安装 Eclipse 的操作步骤如下:

(1) 解压缩 tar-zxvf(Eclipse 安装文件)。

(2) 进入安装目录,直接运行 Eclipse,如图 2.35 所示。

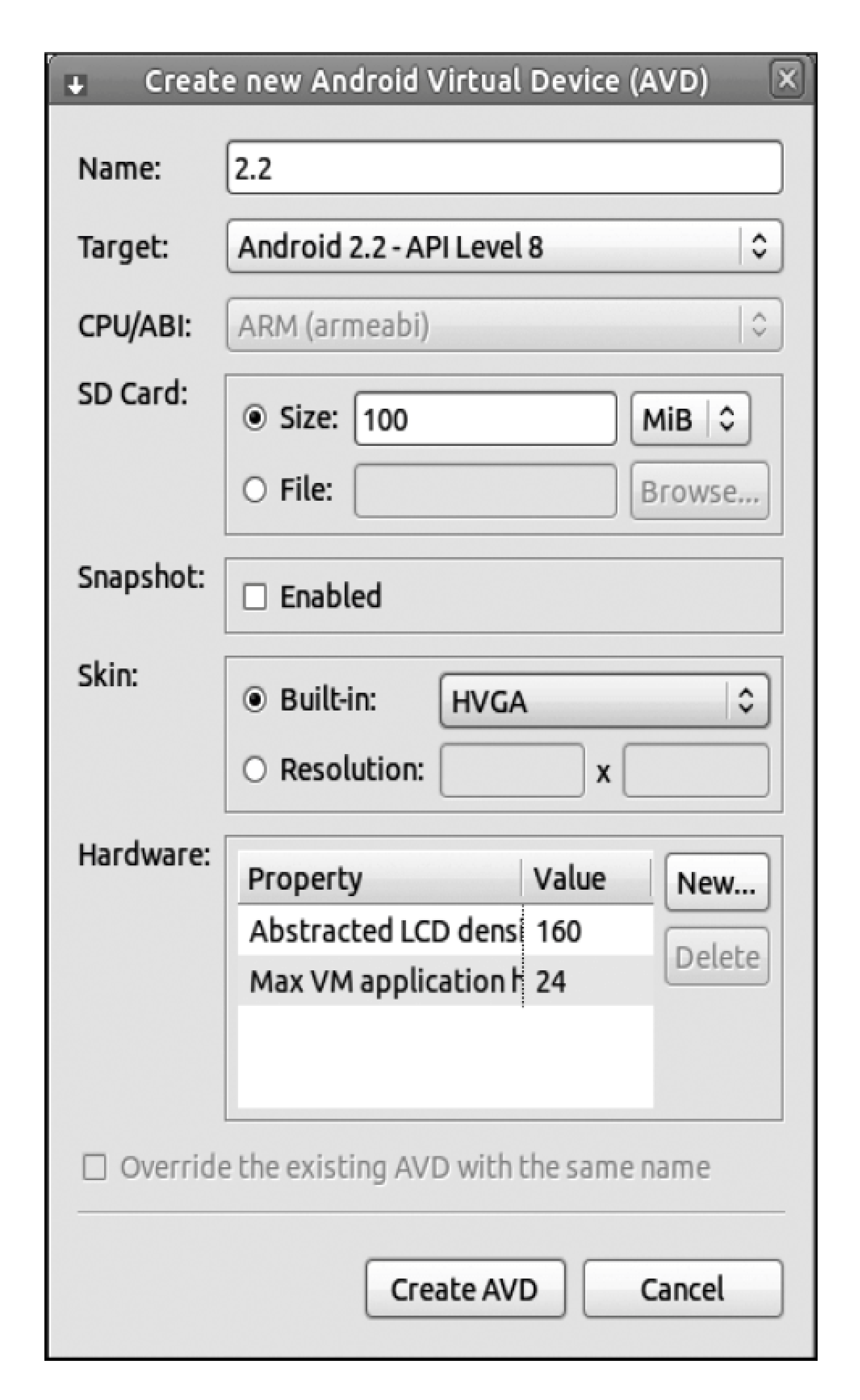


图 2.34　Create New AVD

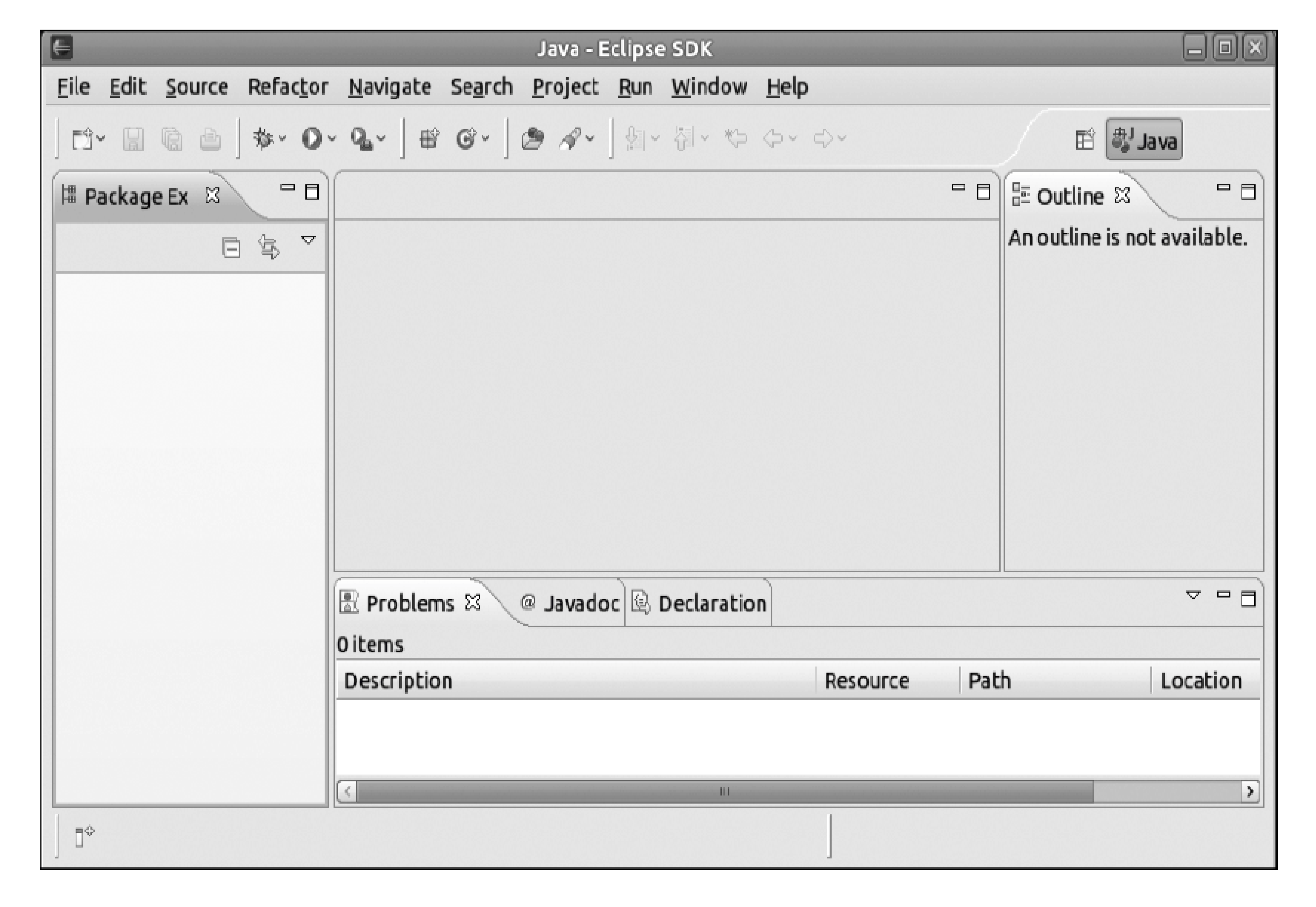


图 2.35　Eclipse 主界面

(3) 安装 ADT 插件。进入 Eclipse，选择菜单 Help→Install New Software，弹出如图 2.36 所示的对话框。

在 Work with：框中输入“https://dl-ssl.google.com/android/eclipse/”，按照提示完

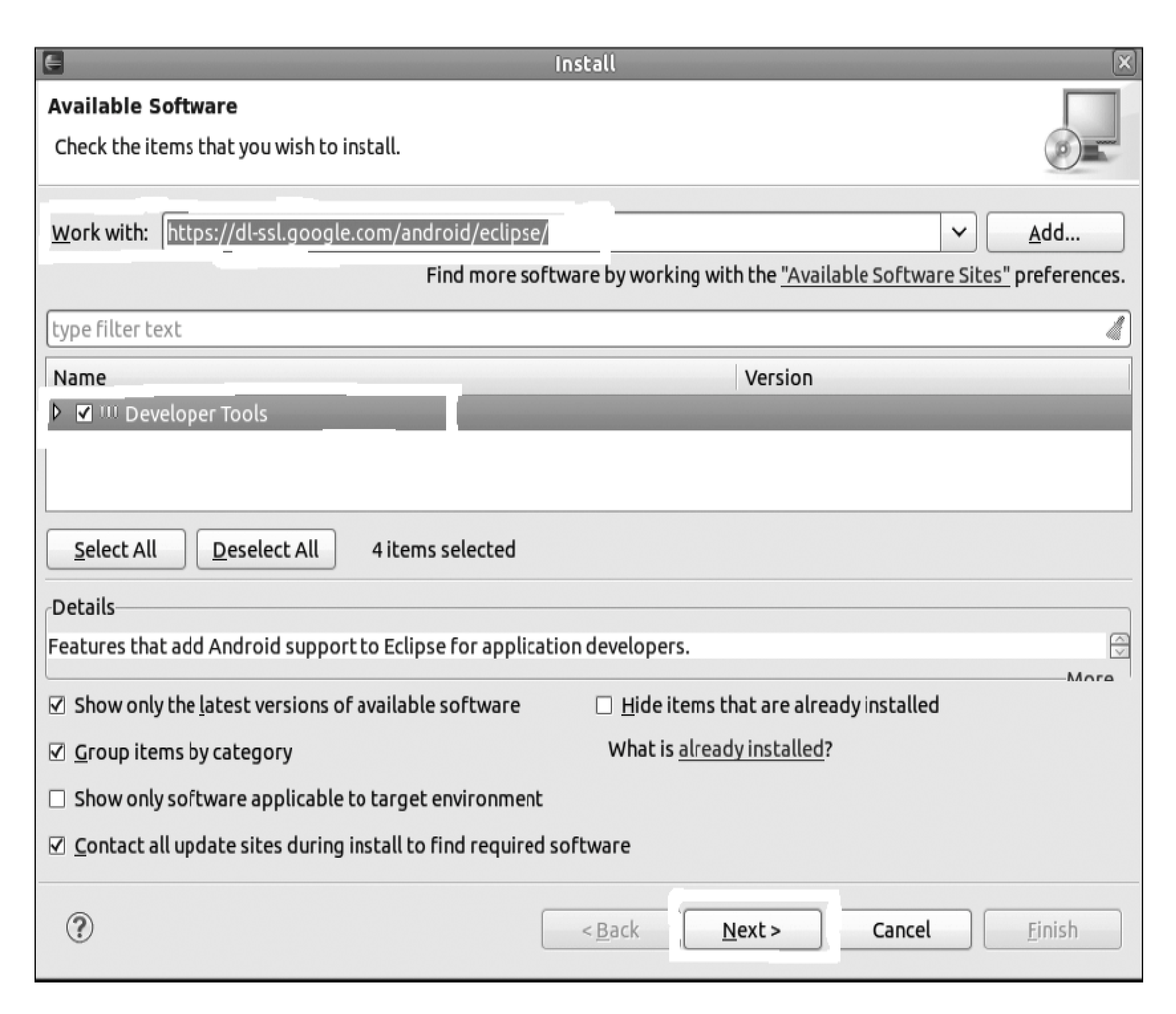


图 2.36　Install New Software

成安装。

（4）设置 Android ADT。进入 Eclipse，选择菜单 Window→Preferences，弹出如图 2.37 所示的对话框。

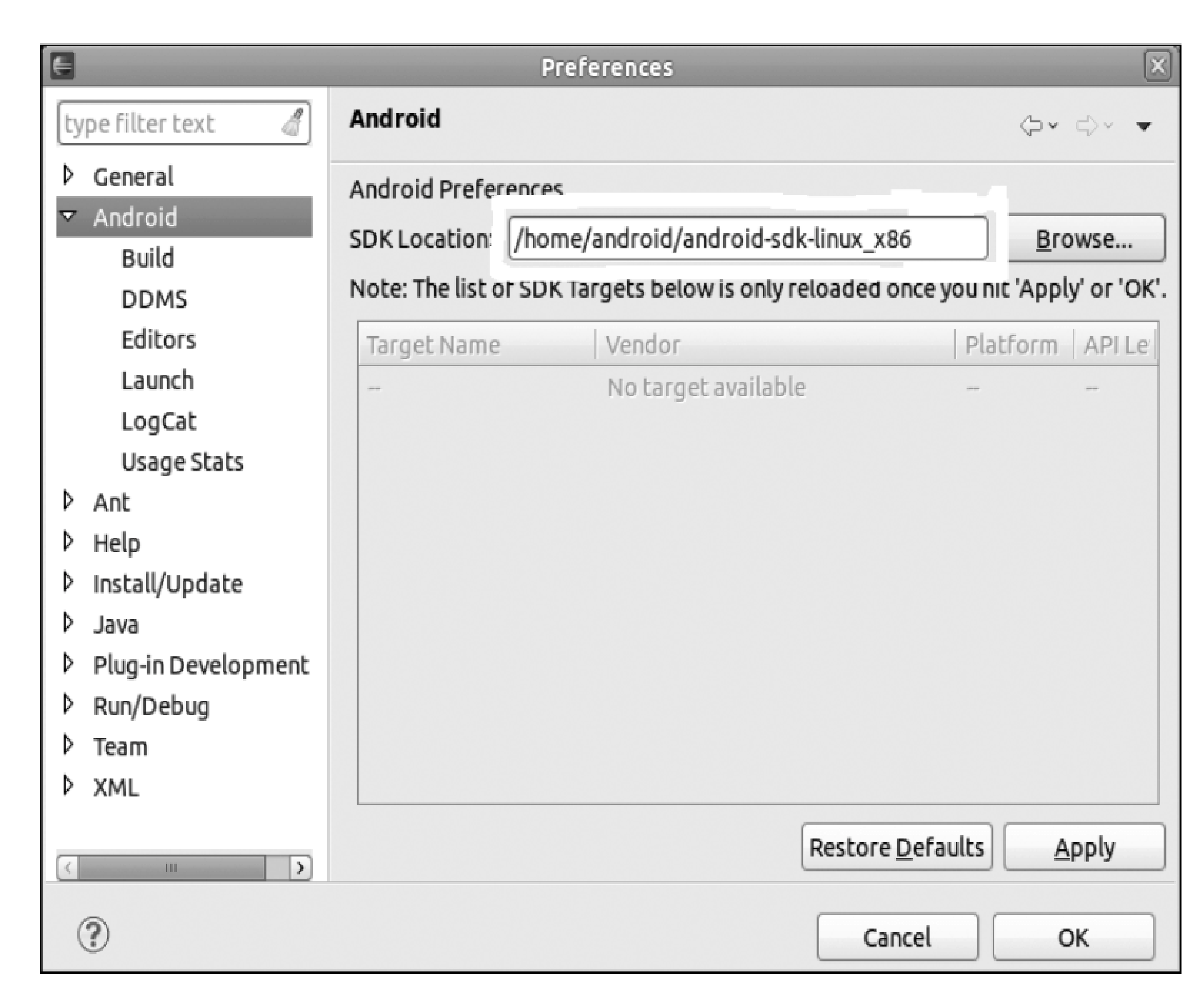


图 2.37　Preferences

单击左边栏的 Android，在右边栏中的 SDK Location 文本框中输入 Android SDK 的安装路径，然后单击 Apply 按钮。

2.5 第一个 Android 程序——Hello World

下面开始创建第一个 Android 项目——Hello World，开发环境是 Windows＋Eclipse。

2.5.1 创建 Android 项目

创建 Android 项目的操作步骤如下。

选择菜单 File→New→Android Application Project，弹出如图 2.38 所示的对话框。

图 2.38 New Android Application

如果没有看到此菜单项，选择菜单 File→New→Other，在如图 2.39 所示的对话框中选择。

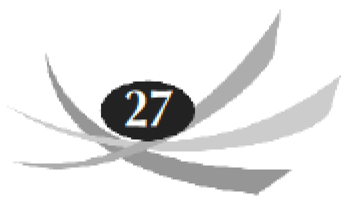
如图 2.40 所示，Application Name、Project Name、Package Name 自定义，Mininum Required SDK 选择最低支持的 Android 版本，Target SDK、Compile With 选择当前需要支持的 Android 版本。维护完成之后单击 Next 按钮，弹出如图 2.41 所示的对话框。

在此处设置目录结构，单击 Next 按钮，弹出如图 2.42 所示的对话框。

在此处设置图标的相关参数，单击 Next 按钮，弹出如图 2.43 所示的对话框。

在此处设置创建的 Activity 类型，Blank Activity 支持的版本较广。单击 Next 按钮，弹出如图 2.44 所示的对话框。

在此处设置默认创建的页面的 Activity 相关信息，单击 Finish 按钮创建完成。



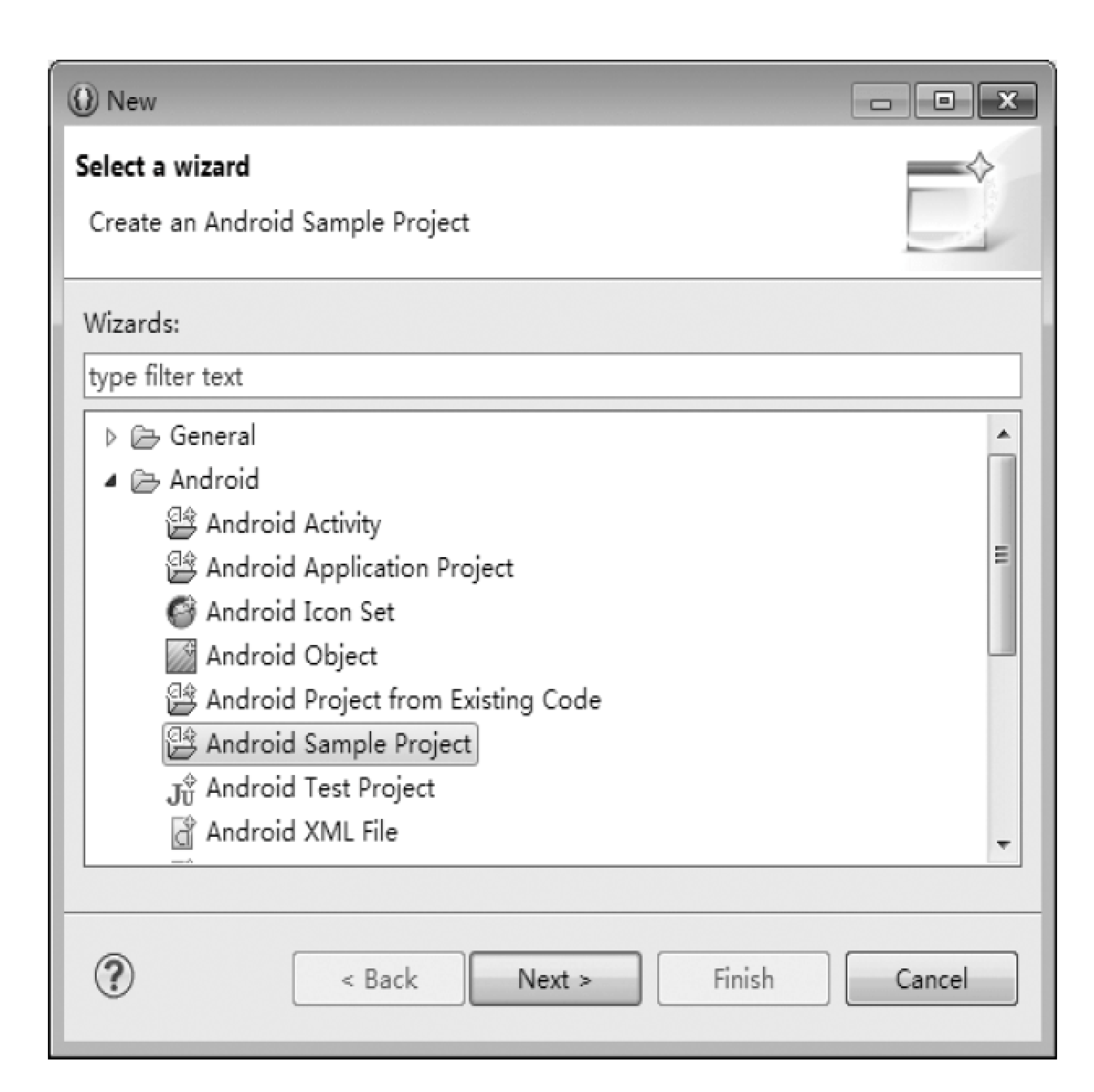


图 2.39　Select a wizard

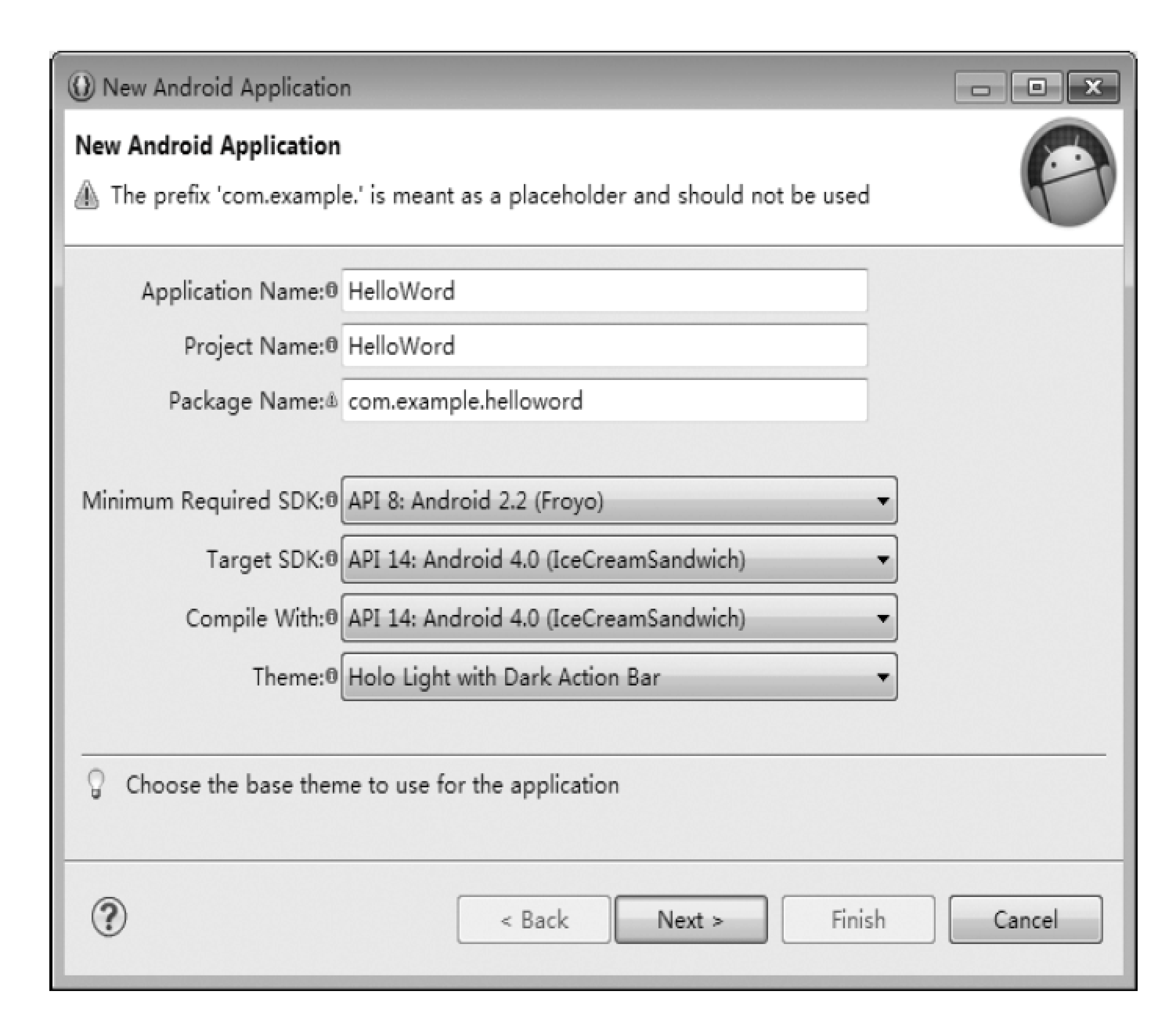


图 2.40　New Android Application



图 2.41 New Android Application

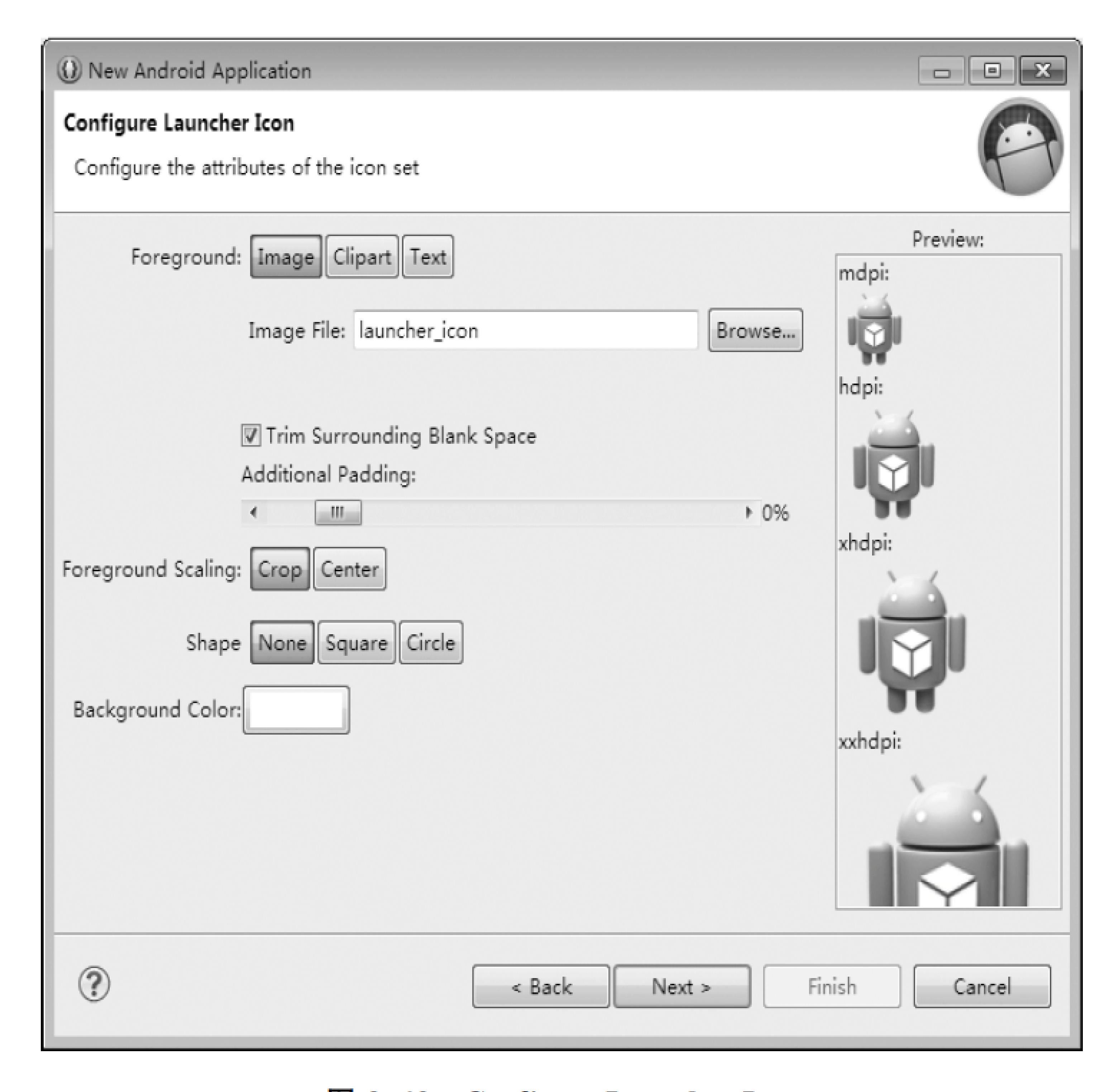


图 2.42 Configure Launcher Icon



图 2.43　Create Activity



图 2.44　Blank Activity

2.5.2　项目目录结构

在 Eclipse 中展开项目文件,文件结构如图 2.45 所示。

1. src

在 src 下是主程序类。如果在建立项目时,选择并填写了 Create Activity 时,会自动生成名为填写内容的,继承自 Android.app.Activity 的类。在类中重写了 onCreate()方

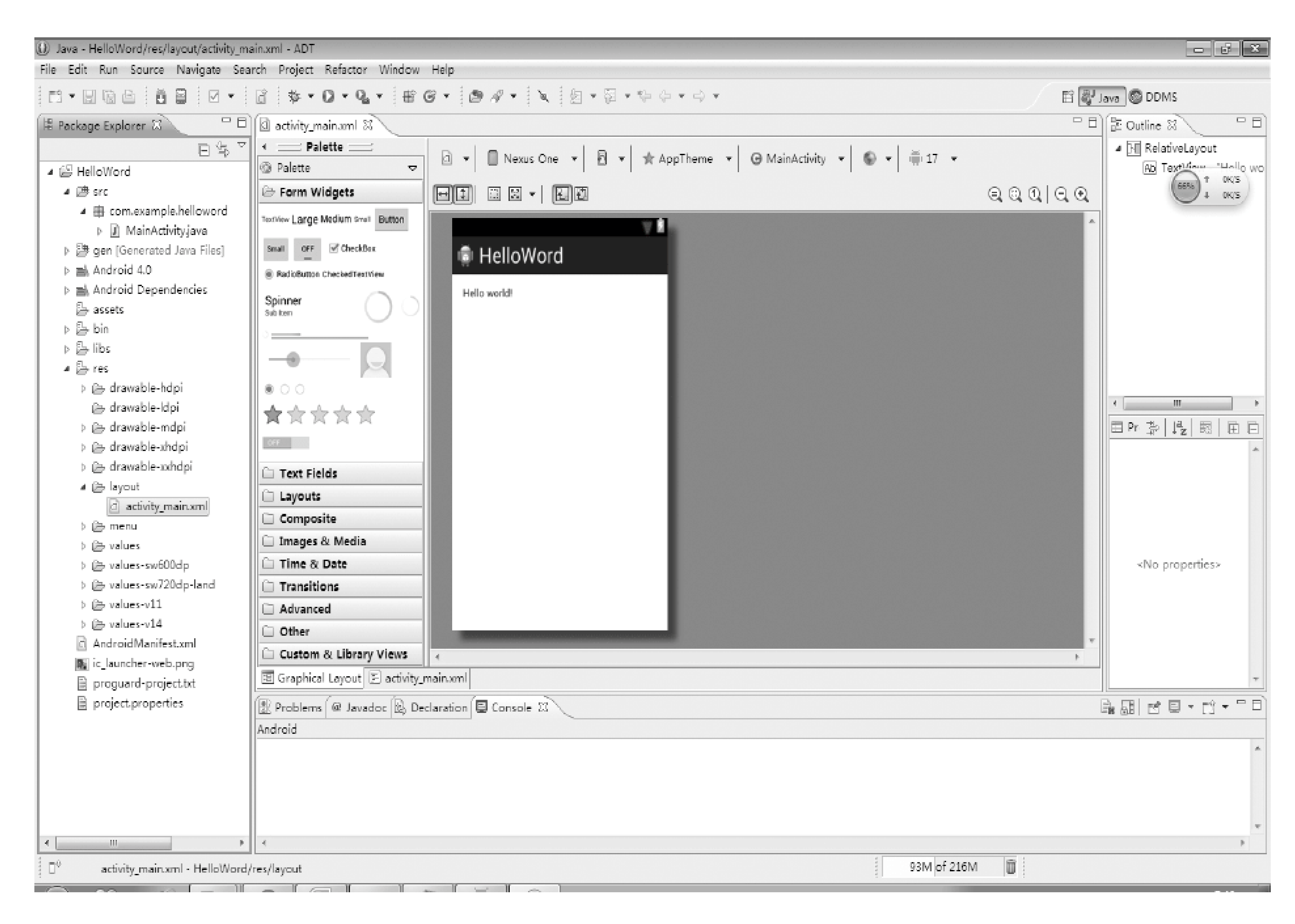


图 2.45 Eclipse 目录结构

法。方法中的 setContentView 为设置这个 Activity 的显示布局(R. layout. main),布局文件在 res/layout 下。R. layout. main 实际上是指 res/layout/main. xml 布局文件。

Java 代码如下:

```
import Android.os.Bundle;
import Android.app.Activity;
import Android.view.Menu;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

2. gen

gen 中的文件为编译器自动生成的。R 文件是放在 res 中的元素的列表,元素会在 R 中自动生成一个 ID,在程序中可以直接引用 ID 来获取元素。使用 aidl 通信模式时,也会在 gen 中生成对应的类。

3. res

在 res 下为资源文件,包括 res/drawable(png、jpg 格式图片)、res/layout(View 对象的 XML 表示)、res/values(字符串 String、颜色 color、样式 style、尺寸 dimen、数组 array 的 xml 表示)、res/menu(菜单)。

4. assets

asset 是指程序其他的资源文件,例如 html 文件、js 文件、图片文件等其他文件。

WebView 加载 html 文件时,URI 地址这样写:

```
webView.loadUrl("file:///Android_asset/index.html");
```

5. AndroidMainifest 配置文件

每个项目都有 AndroidMainifest.xml 配置文件,例如,如下 XML 代码:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:Android="http://schemas.Android.com/apk/res/Android"
    package="com.example.helloword"
    Android:versionCode="1"
    Android:versionName="1.0" >
    <uses-sdk
        Android:minSdkVersion="8"
        Android:targetSdkVersion="14" />
    <application
        Android:allowBackup="true"
        Android:icon="@drawable/ic_launcher"
        Android:label="@string/app_name"
        Android:theme="@style/AppTheme" >
        <activity
            Android:name="com.example.helloword.MainActivity"
            Android:label="@string/app_name" >
            <intent-filter>
                <action Android:name="Android.intent.action.MAIN" />
                <category Android:name="Android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

注意:加底纹的两行在 AndroidMainifest.xml 中必须有,而且只能出现一次。

首先是 manifest 元素的声明,包括包声明和 Android 的命名空间。然后是 application,定义了图标和名称属性,可以定义 name 属性,使用扩展 Application 的对象存储屏幕之间共享的数据。

接下来是 application 的子元素 Activity、Service 或 Receiver,如表 2-1 所示。

表 2-1　application 子元素

元　　素	位　　置	说　　明
<manifest>	root	定义应用程序的包名和 Android 命名空间
<users-permission>	root	请求开放一个安全权限，例如开启 GPS、收发短信、添加访问 WIFI 及网络权限等
<permission>	root	声明一个安全权限
<instrumentation>	root	声明一个检测设备组件
<uses-sdk>	root	sdk 最低兼容版本
<application>		
<activity>	<application>子元素	
<service>	<application>子元素	
<receiver>	<application>子元素	
<provider>	<application>子元素	
<uses-library>	<application>子元素	引用其他包时，需要声明
<intent-filter>	<activity，service，receiver>子元素	
<action>	<intent-filter>子元素	intent 动作
<category>	<intent-filter>子元素	intent 类别
<data>	<intent-filter>子元素	intent 的 MIME 类型、URI 方案、URI 授权、URI 路径

application 配置非常重要，诸如 Activity、权限、Intent 等都在这里配置。

- package：定义该应用的包。
- Android：versionCode：定义应用的版本号。
- Android：versionName：定义应用的版本名字。
- application 标签：定义一个应用，一个项目最多有一个 Application 标签。
- Android：icon="@drawable/icon"：定义应用的图标引用资源文件中的 icon 图片。
- Android：label="@string/app_name"：定义应用的名称。
- activity 标签：定义一个 Activity，每一个 Activity 都必须在这里定义，否则不能运行。
- Android：name：定义 Activity 的类名，这里的 HelloWorld 是以上面的 Package 定义为基础的，也就是 Package(com. flysnow)加上这个 Android：name(. HelloWorld)要能定位到这个 Activity(com. flysnow. HelloWorld)，否则就找不到。
- Android：label：定义该 Activity 的标题。
- intent-filter：定义一个 Intent 过滤器，用于标记对应的 Activity，以便 Android 系统能找到这个 Activity，定义的是隐性的 Intent，主要使用两个子标签 action 和 category 来区分每个 Intent。

• ＜uses-sdk Android：minSdkVersion="8"/＞：定义应用的最低 SDK 的级别。

2.5.3 运行项目

右击项目，在弹出的快捷菜单中选择 Run as→Run Configuration 命令，弹出如图 2.46 所示的对话框。

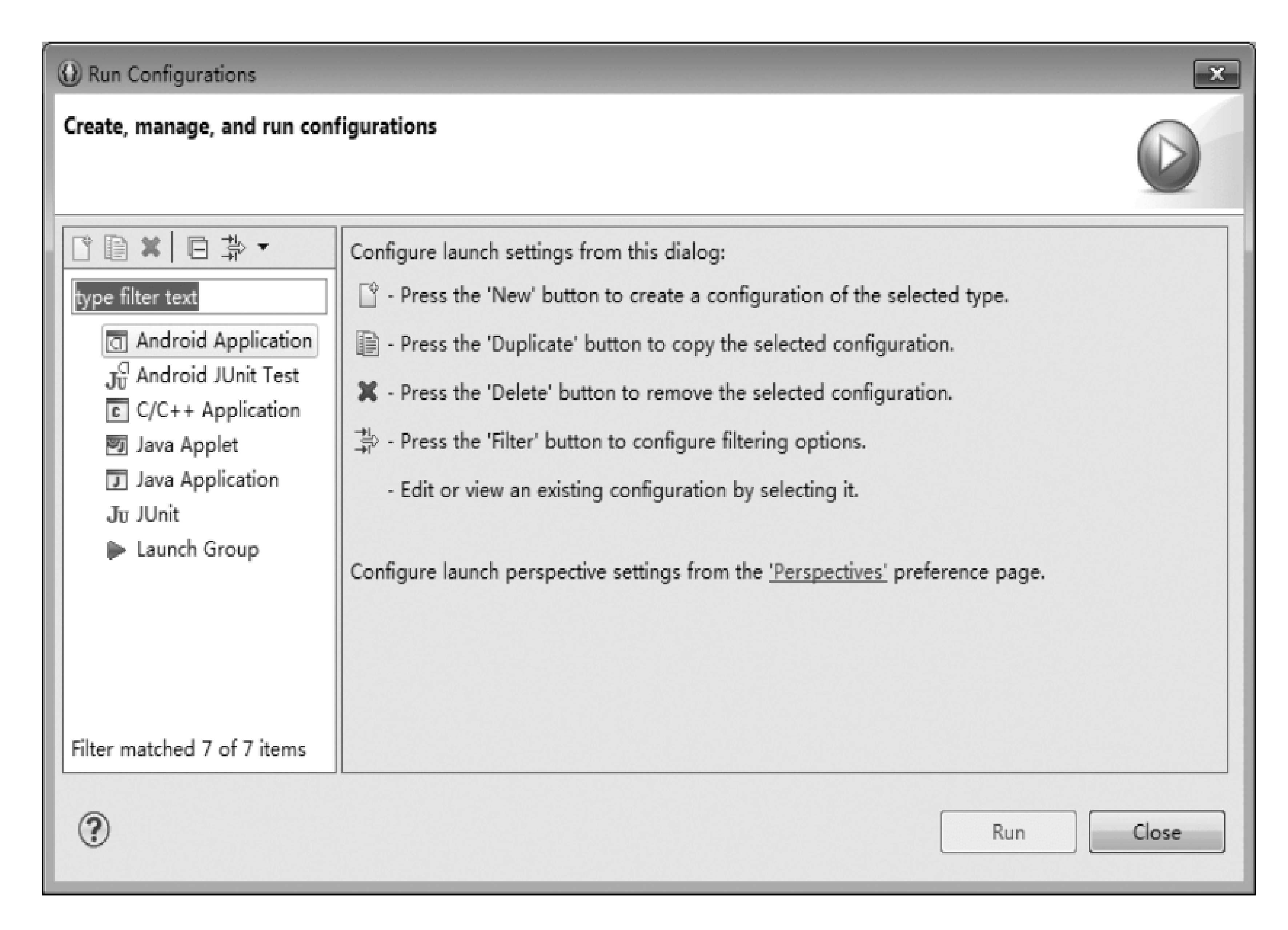


图 2.46 **Run Configurations(1)**

双击 Android Application，弹出如图 2.47 所示的对话框。

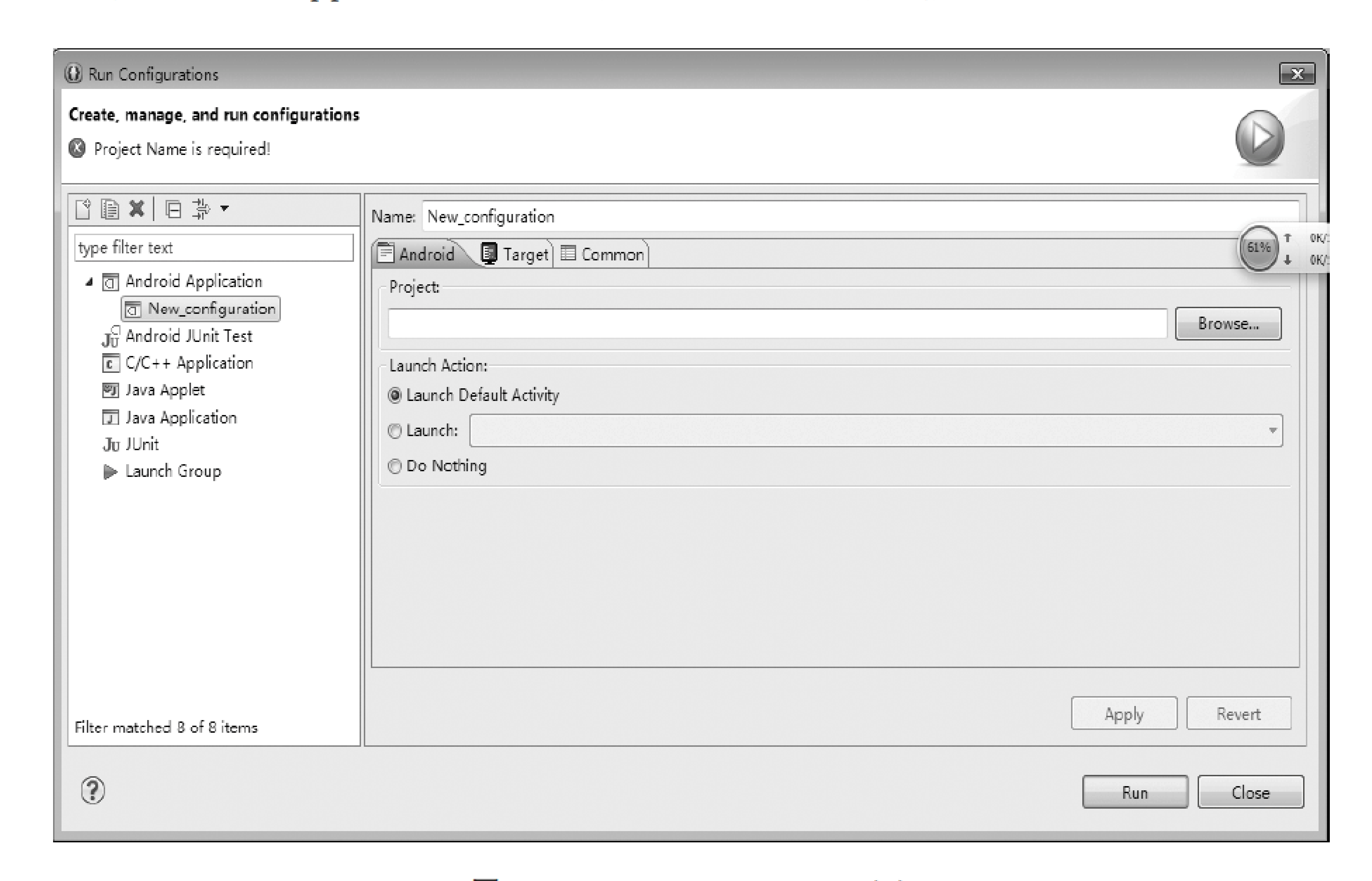


图 2.47 **Run Configurations(2)**

单击 Browse 按钮，选择要运行的项目。在 Name 文本框中输入名称“HelloWorld”(可以自定义)。

单击 Target 选项卡，切换到如图 2.48 所示的对话框。

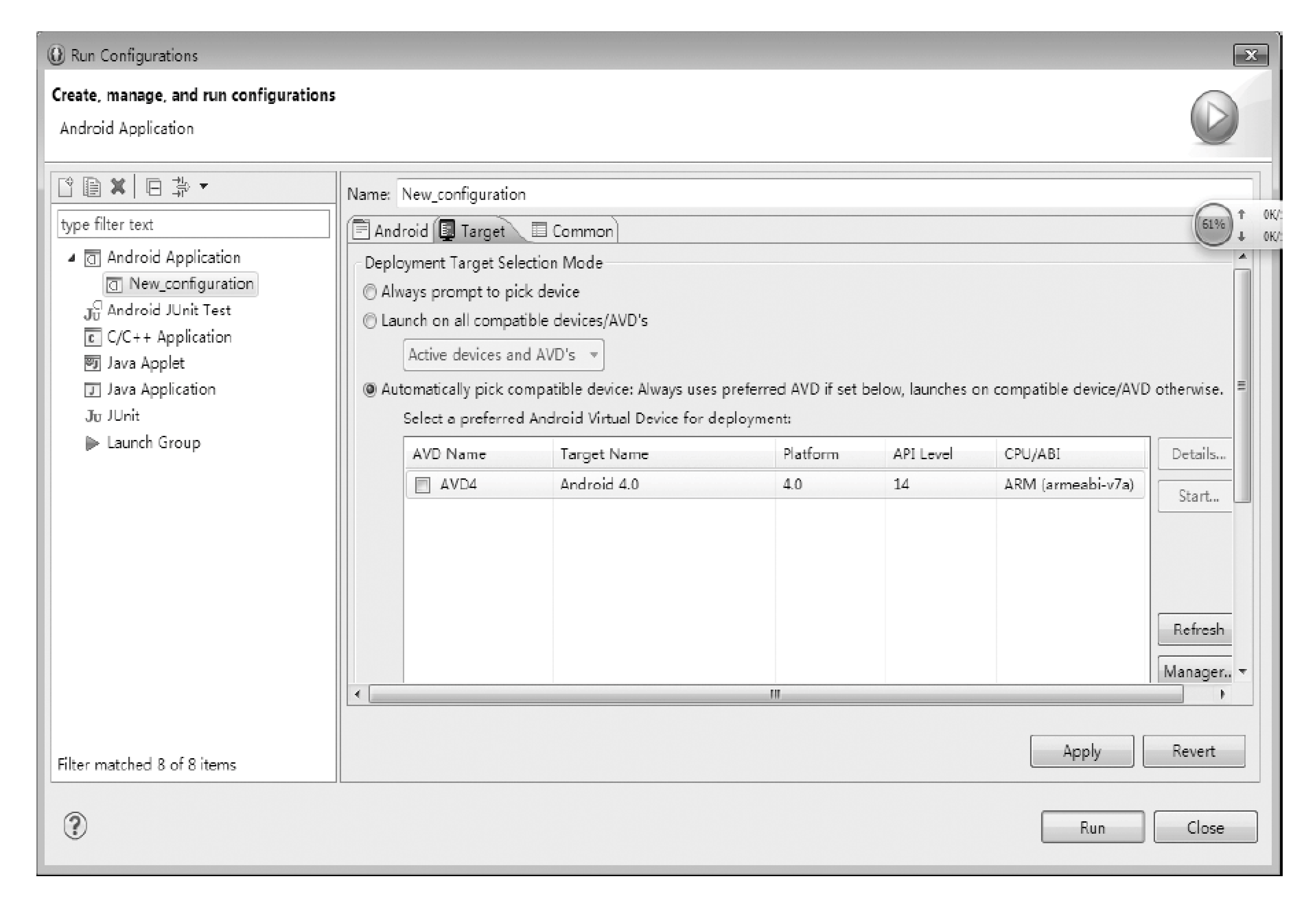


图 2.48　Run Configurations(3)

选择运行的 AVD，单击 Apply 按钮保存上面的设置。单击 Run 按钮或者右击项目名称，从弹出的快捷菜单中选择 run as→Android Application 命令，即可启动该 Android 程序，自动弹出 Android 虚拟机界面，如图 2.49 所示。



图 2.49　Android 虚拟机

启动完成后，解锁 Android 系统，即可显示 HelloWorld 界面，如图 2.50 所示。同时，在 Eclipse 的 Console 界面显示如图 2.51 所示。

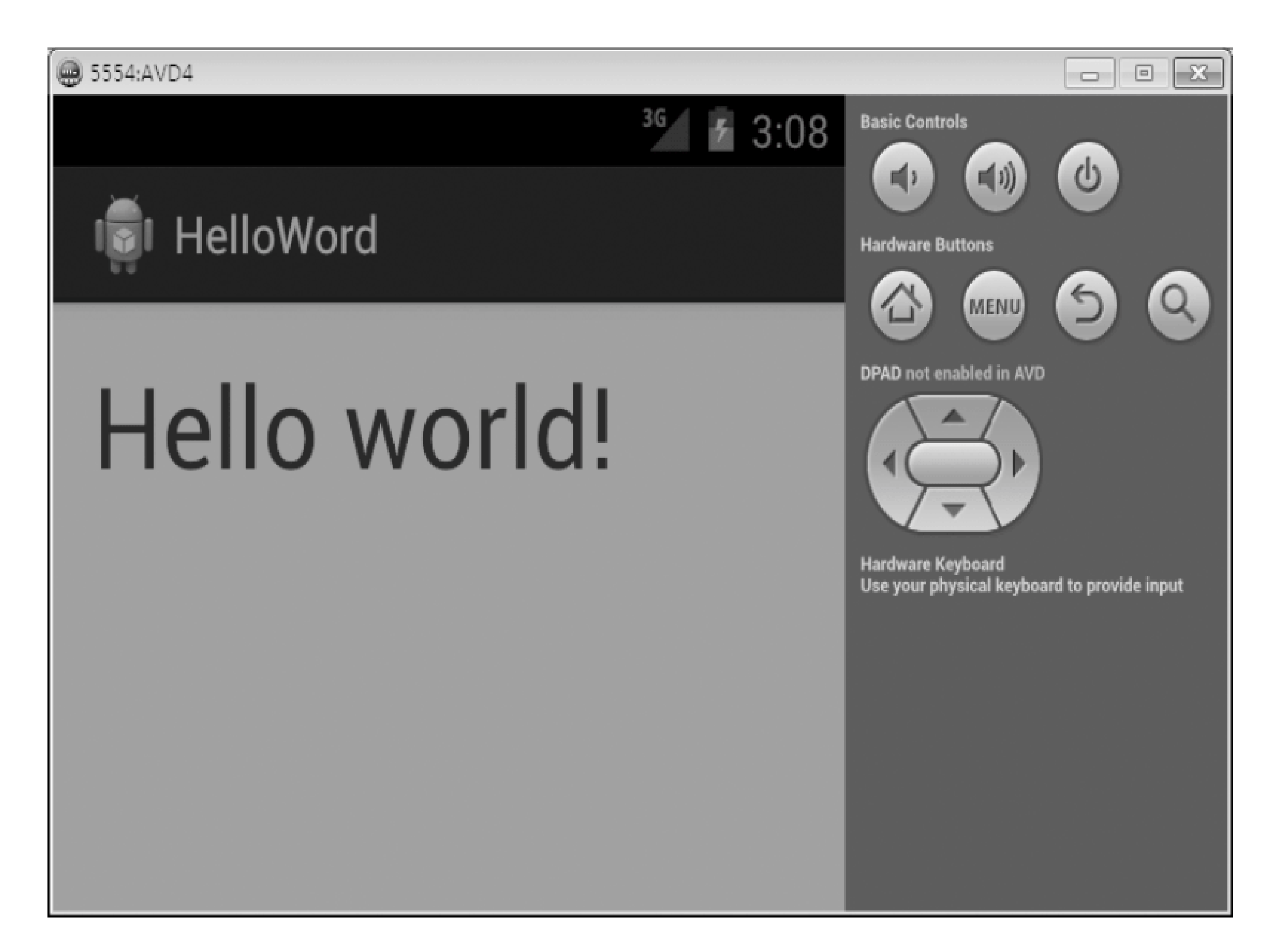


图 2.50 HelloWorld 界面

```
Problems  Javadoc  Declaration  Console  LogCat
Android
[2013-03-03 08:11:02 - HelloWord] ------------------------------
[2013-03-03 08:11:02 - HelloWord] Android Launch!
[2013-03-03 08:11:02 - HelloWord] adb is running normally.
[2013-03-03 08:11:02 - HelloWord] Performing com.example.helloword.MainActivity activity launch
[2013-03-03 08:11:02 - HelloWord] Automatic Target Mode: Preferred AVD 'AVD4' is not available. Launching new emulator.
[2013-03-03 08:11:02 - HelloWord] Launching a new emulator with Virtual Device 'AVD4'
[2013-03-03 08:11:10 - Emulator] Failed to create Context 0x3005
[2013-03-03 08:11:10 - Emulator] emulator: WARNING: Could not initialize OpenglES emulation, using software renderer.
[2013-03-03 08:11:10 - Emulator] could not get wglGetExtensionsStringARB
[2013-03-03 08:11:10 - Emulator] could not get wglGetExtensionsStringARB
[2013-03-03 08:11:10 - Emulator] could not get wglGetExtensionsStringARB
[2013-03-03 08:11:10 - Emulator] could not get wglGetExtensionsStringARB
[2013-03-03 08:11:10 - Emulator] could not get wglGetExtensionsStringARB
[2013-03-03 08:11:10 - Emulator] could not get wglGetExtensionsStringARB
[2013-03-03 08:11:10 - Emulator] could not get wglGetExtensionsStringARB
[2013-03-03 08:11:10 - Emulator] could not get wglGetExtensionsStringARB
[2013-03-03 08:11:12 - HelloWord] New emulator found: emulator-5554
[2013-03-03 08:11:12 - HelloWord] Waiting for HOME ('android.process.acore') to be launched...
[2013-03-03 08:13:12 - HelloWord] HOME is up on device 'emulator-5554'
[2013-03-03 08:13:12 - HelloWord] Uploading HelloWord.apk onto device 'emulator-5554'
[2013-03-03 08:13:12 - HelloWord] Installing HelloWord.apk...
[2013-03-03 08:15:05 - HelloWord] Success!
[2013-03-03 08:15:05 - HelloWord] Starting activity com.example.helloword.MainActivity on device emulator-5554
[2013-03-03 08:18:29 - HelloWord] ------------------------------
```

图 2.51 Eclipse Console 窗口

2.6 Android 测试

Android 测试有多种方法：第一种是使用 Eclipse 的 Debug 运行方式；第二种是使用 Android 的 Log 类记录相应的测试信息。下面主要介绍 Log 类。

2.6.1 Log 类和方法

Log 类在 Android 的类包 Android.util.Log 中。Log 类有 5 种不同级别：verbose、debug、info、warn 和 error，对应 5 种方法：Log.v(tag,msg)、Log.d(tag,msg)、Log.i(tag,msg)、Log.w(tag,msg)和 Log.e(tag,msg)。

- 参数 tag：用于标识日志消息的来源。它通常用于标识类或 Activity 日志调用出现的位置。在 LogCat 窗口中可以使用定义的 LOG_TAG 来过滤所有使用这个

TAG 的 log。

- 参数 msg：log 的信息。

示例如下：

```
public static final String LOG_TAG= "com.activity.example";
Log.v(CommonConfig.LOG_TAG, "ActivityExample onCreate");
```

2.6.2 LogCat 页面

选择菜单 Window→Show View→Other→Android→LogCat 命令，打开 LogCat 页面，如图 2.52 所示。

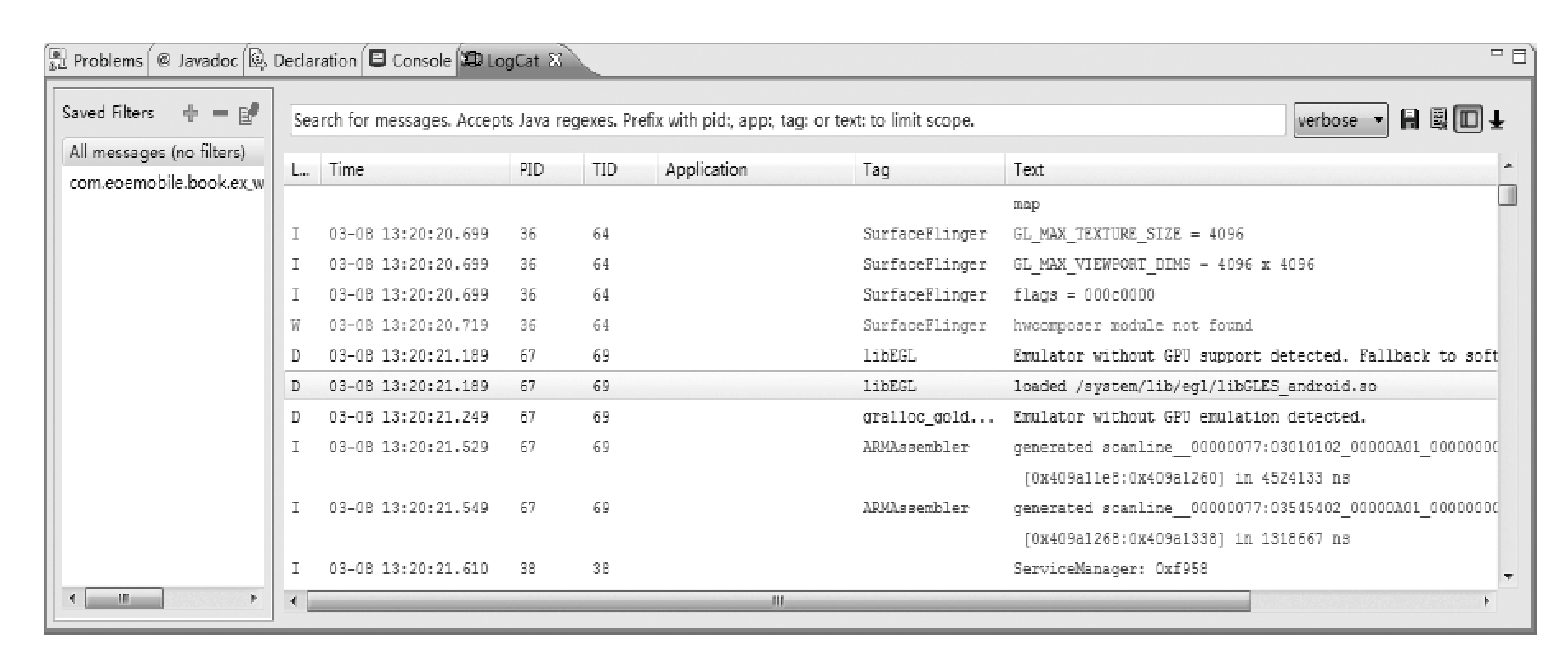


图 2.52 LogCat 页面

在这个页面可以看到系统运行中所有的日志信息，其中包括我们使用 Log 类记录的测试信息。

为方便查找测试信息，可以加一个过滤。单击页面左上角的加号，如图 2.53 所示对话框，在 Filter Name 文本框中输入要创建的 Filter 的名字（自定义），在 by Application Name 文本框中输入要过滤的信息的 Application，再根据需要维护其他过滤信息，单击 OK 按钮即可。

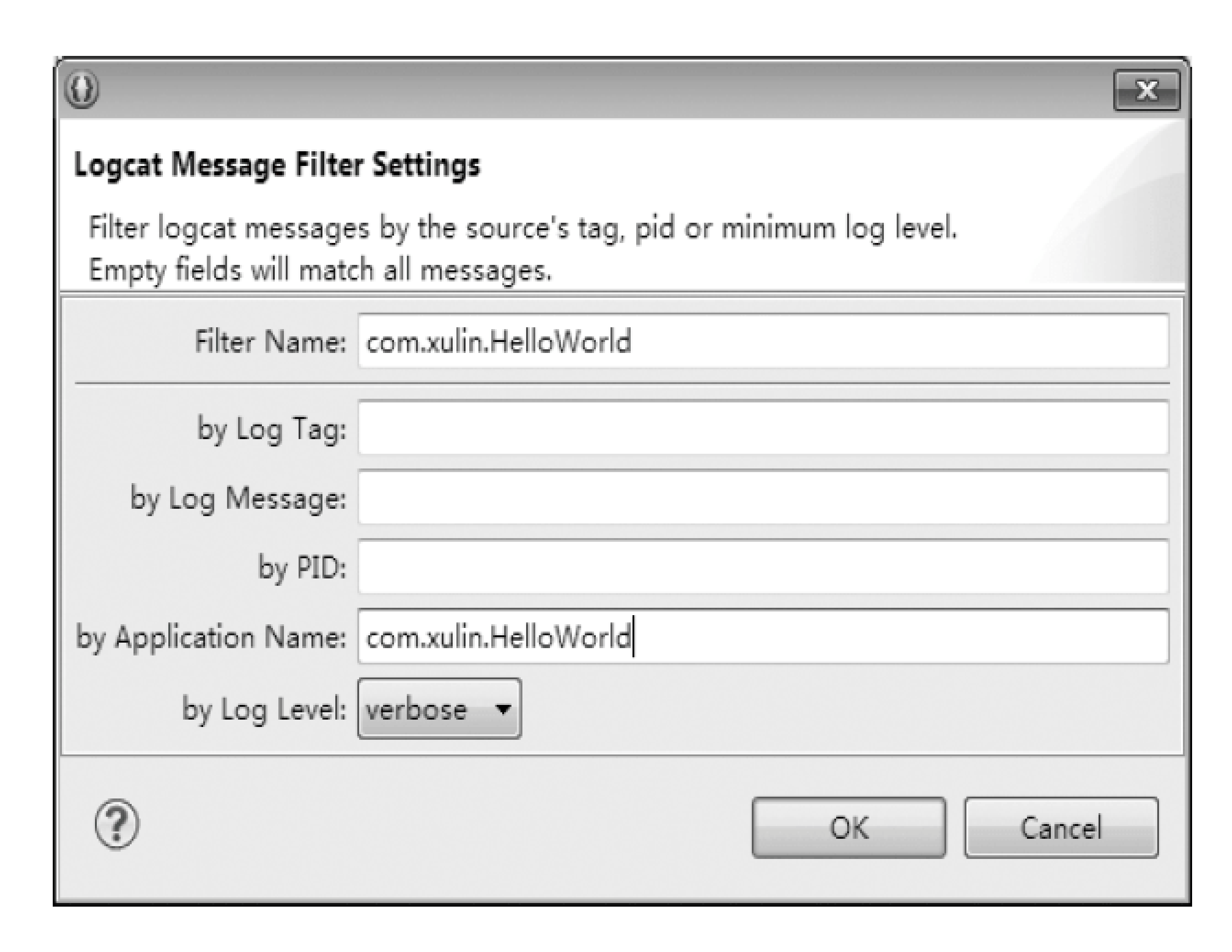


图 2.53 Logcat Message Filter Settings 对话框

2.7　Activity

Activity 是 Android 应用程序的入口，负责创建窗口(setContextView(View))以及和用户交互等。

2.7.1　基本用法

首先定义一个类继承自 Android. app. Activity，并在 AndroidManifest. xml 文件的 application 标签中声明该 activity 即可。

一般一个程序可能会有多个 Activity，需要指定程序运行时首先加载的 Activity，在 activity 标签内部需要定义 action 和 category。

2.7.2　常用设置

AndroidManifest 中的通常设置如表 2-2～表 2-4 所示。

表 2-2　Activity 通常设置

设　　置	名　称	说　　明
Android：launchMode	启动模式	standard，singleTop，singleTask，singleInstance
Android：screenOrientation	屏幕	landscape(横屏)，portrait(竖屏)
Android：label	标题名称	直接写字符，或引用 xml 文件中的@string/
Android：name	Activity 类名	注意：一般是 manifest 的 package 加 name 等于 Activity 类带包名的全称

表 2-3　Java 常用方法

方　　法	功　　能	说　　明
setContentView()	设置 Activity 视图	参数可以是：①res/layout 中的 xml 文件，通过 R. layout 获取 int 的 id；②View 类型的视图
findViewById	在 XML 文件中寻找 View	R 中的 id。一般设置 setContentView 为 R 中的 layout，通过这个方法寻找每个组件的引用

表 2-4　Java 其他方法

方　　法	说　　明
去标题	super. requestWindowFeature(Window. FEATURE_NO_TITLE)需要在执行 setContentView 之前操作，否则报错
全屏	super. getWindow(). setFlags(WindowManager. LayoutParams. FLAG_FULLSCREEN，WindowManager. LayoutParams. FLAG_FULLSCREEN)；
横竖向	super. setRequestedOrientation(ActivityInfo. SCREEN_ORIENTATION_LANDSCAPE)；/ ActivityInfo. SCREEN_ORIENTATION_PORTRAIT
屏幕参数 widht、height、dpi 等	Display display＝super. getWindowManager(). getDefaultDisplay()； DisplayMetrics displayMetrics＝new DisplayMetrics()； display. getMetrics(displayMetrics)；display. getWidth()、displayMetrics. densityDpi 等方法

2.7.3 生命周期

Activity 的生命周期如图 2.54 所示，Activity 有 7 个生命周期。Activity 程序入口为方法 onCreate 或 onStart。一些初始化的操作需要在这两个方法中进行，例如设置 layout、初始化控件、添加事件监听等。

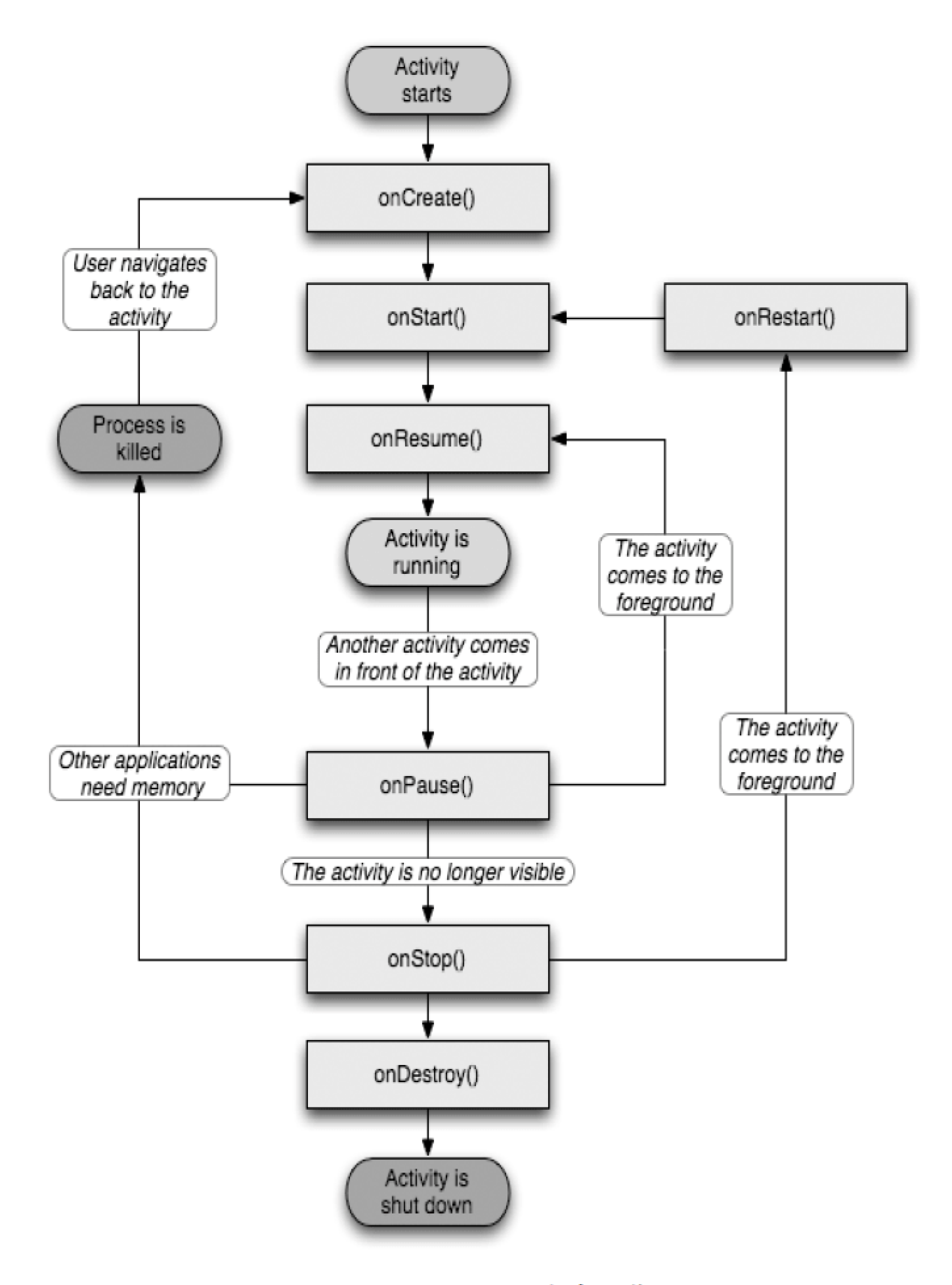


图 2.54 Activity 生命周期

每次启动 Activity 都是从 OnCreare 开始，接着是 onStart 和 onResume。

按 Back 键就结束程序：onPause，onStop，onDestroy。

按 home 键切换程序：onPause，onStop，切换回来：onRestart，onStart，onResume。

在两个以上 Activity 切换时，被替换的 Activity 是否被 Destroy 取决于 Android 加载模式。

Activity 方法如表 2-5 所示。

2.7.4 Activity 加载模式

在配置文件中 Android：launchMode 可以配置 Activity 加载模式。Activity 的加载模式有 4 种：standard、singleTop、singleTask 和 singleInstance。

表 2-5 Activity 方法

方法	说 明
onCreate	当首次创建 Activity 时调用。一些设置在该方法中进行：创建视图、数据绑定等。还以 Bundle 的形式提供以前存储的任何状态的访问
onRestart	重新启动 Activity 时调用。该活动仍在内存中，而不是重起一个新的 Activity
onStart	当 Activity 在屏幕上可见时调用。在 onResume()之后被调用
onResume	当 Activity 与用户交互时调用
onPause	在系统启动恢复前一个活动时调用。这通常用于未保存的更改提交到持久性数据、停止动画和其他可能会占用 CPU 等的东西。此方法的实现必须非常快速，因为此方法返回之前不会恢复到下一个活动。如果返回到前一个活动，跟在 OnResume()后面；如果对用户不可见，跟在 onStop()后面
onStop	因为另一项 Activity 已恢复或启动，之前的 Activity 将不可见。这可能是因为正在启动一个新的 Activity，之前的这一个 Activity 或被销毁。如果之前的这个 Activity 出错或处理失败，将调用 onDestroy()；如果被用户重新选中，则调用 onRestart()
onDestroy	Activity 被销毁前的最后调用。发生这种情况是因为 Activity 被 finishing(或称为 finish())，或因为系统暂时销毁此活动的 Activity 以节省空间。可以使用 isFinishing()方法区分这两种情况

(1) standard：默认，表示如果需要就创建。这样会导致切换一次 Activity，就创建一个。切换多少次，按 back 键就会返回多少次。

例如，ActivityMain 使用 standard 模式，当它转到 ActivityOtherOne，又切换到 ActivityMain 时，原先在 ActivityMain 上面的 EditText 最开始输入的文字也就没有了。

(2) singleTop：如果已经有一个 Activity 实例位于 Activity 栈的顶部时，就不产生新的实例。

(3) singleTask：会在一个新的 task 中产生这个实例，以后每次调用都会使用这个，不会产生新的实例了。

(4) singleInstance： 这个跟 singleTask 基本上是一样，只有一个区别：在这个模式下的 Activity 实例所处的 task 中，只能有这个 Activity 实例，不能有其他的实例。

2.7.5 Activity 切换

首先简单介绍 Intent(意图)。Intent 类相当于平台中应用程序之间的通信网络，Intent 是一个要执行的操作的抽象说明，相当于各个 Activity 之间的桥梁。在一个 Activity 中想切换到另一个 Activity，需要使用方法 startActivity，并需要定义一个 Intent 来指定意图的组件类。

1. 最简单的 Activity 切换

定义 Intent，使用 Intent 设置需要切换到哪个 Activity。使用 Activity 的 startActivity(Intent)方法如下：

```
Intent intent= new Intent(this, ActivityOtherOne.class);
super.startActivity(intent);
```

```
Intent intent= new Intent(this, ActivityOtherOne.class);
super.startActivity(intent);
```

2. 使用 Intent 传递数据

当然也可以在切换 Activity 的时候传递数据。可以直接使用 intent 的 putExtra 方法,也可以新建一个 Bundle 传输入,但是需要让 intent 把 bundle 进行 putExtras 下。

传参数端的 Activity 如下:

```
Intent intent= new Intent(this, ActivityOtherOne.class);
Bundle bundle= new Bundle();
bundle.putString("value1", "hello");
intent.putExtras(bundle);
intent.putExtra("value2","world");
super.startActivity(intent);
Intent intent= new Intent(this, ActivityOtherOne.class);
Bundle bundle= new Bundle();
bundle.putString("value1", "hello");
intent.putExtras(bundle);
intent.putExtra("value2","world");
super.startActivity(intent);
```

接收方,例子中的 ActivityOtherOne 需要在 OnCreate 方法中获取:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
.
.
.
String value1= super.getIntent().getExtras().getString("value1");
String value2= super.getIntent().getExtras().getString("value2");
this.myEditText.setText("value1: "+ value1+ ", value2: "+ value2);
}
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
.
.
.
    String value1= super.getIntent().getExtras().getString("value1");
    String value2= super.getIntent().getExtras().getString("value2");
    this.myEditText.setText("value1: "+ value1+ ", value2: "+ value2);
}
```

3. 带回传数据的

(1) 在主 Activitiy(ActivityMainExample)需要使用 startActivityForResult 方法切

换 Activity。其中,第二个参数 requestCode 只有当≥0,在传回此 Activity 时才会调用 onActivityResult 方法,得到回传数据。

ActivityMainExample 类中代码如下:

```
Intent intent= new Intent(this, ActivityOtherOne.class);
Bundle bundle= new Bundle();
bundle.putString("value1", "hello");
intent.putExtras(bundle);
intent.putExtra("value2","world");
super.startActivityForResult(intent, 1);
Intent intent= new Intent(this, ActivityOtherOne.class);
Bundle bundle= new Bundle();
bundle.putString("value1", "hello");
intent.putExtras(bundle);
intent.putExtra("value2","world");
super.startActivityForResult(intent, 1);
```

重写 onActivityResult 方法,获取回传数据:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
String result1= data.getExtras().getString("result1");
String result2= data.getExtras().getString("result2");
Log.v(CommonConfig.LOG_TAG, "ActivityExample onActivityResult, requestCode:
"+ requestCode+ ", resultCode:  "+ resultCode);
Log.v(CommonConfig.LOG_TAG, "ActivityExample onActivityResult data- result1:
"+ result1+ ", result2:  "+ result2);
}
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    String result1= data.getExtras().getString("result1");
    String result2= data.getExtras().getString("result2");
    Log.v(CommonConfig.LOG_TAG, "ActivityExample onActivityResult,
    requestCode: "+ requestCode+ ", resultCode: "+ resultCode);
    Log.v(CommonConfig.LOG_TAG, "ActivityExample onActivityResult data-
    result1: "+ result1+ ", result2: "+ result2);
}
```

(2) 在另一个 Activity 插入回传数据,然后 finish。

ActivityOtherTwo 类中代码如下:

```
@Override
public void onClick(View v) {
super.getIntent().putExtra("result1", "hello Android");
super.getIntent().putExtra("result2", "hello google");
super.setResult(Activity.RESULT_OK, super.getIntent());
```

```
super.finish();
}
@Override
public void onClick(View v) {
    super.getIntent().putExtra("result1", "hello Android");
    super.getIntent().putExtra("result2", "hello google");
    super.setResult(Activity.RESULT_OK, super.getIntent());
    super.finish();
}
```

2.7.6 其他常用的 Activity

其他常用的 Activity 如图 2.55 所示。

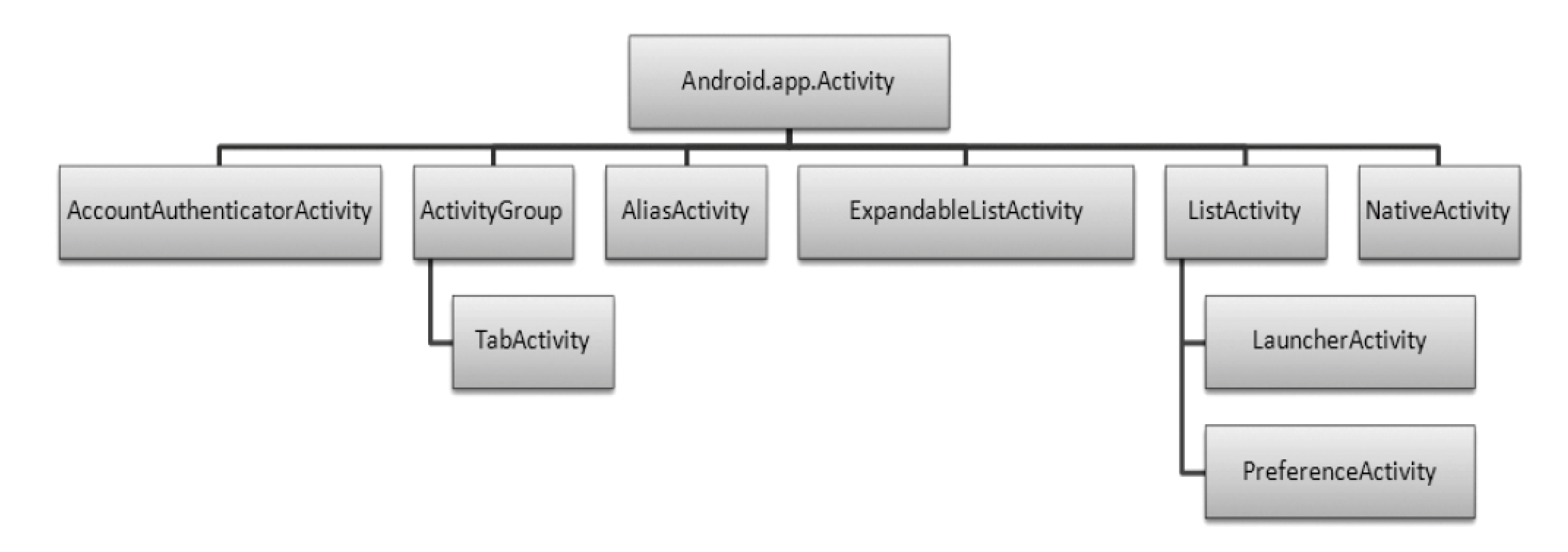


图 2.55 Activity 继承关系

第 3章

基本界面控件

通过第 2 章的学习,我们知道了如何创建一个简单的 Android 项目,如何测试 Android 项目。在现实生活中,面对的系统都是带有操作界面的,界面上有输入框、说明信息、选择框、图片、按钮等控件。本章结合一个信息采集界面讲解最常用的 Android 界面控件。

大多数的 Android 界面控件都在 Android. view 和 Android. widget 包中。它们的父类是 Android. view. View。对话框 Dialog 系列的父类是 Android. app. Dialog。Android 的原生控件一般在 res/layout 下的 xml 文件中声明。在 Activity 中通过使用 super. setContentView(R. layout. 某布局 layout 文件名)来加载 layout。在 Activity 中获取控件的引用需要使用 super. findViewById(R. id. 控件的 ID),接着就可以使用这个引用对控件进行操作,例如添加监听、设置内容等。当然也可以通过代码动态地使用控件。

下面介绍主要的继承关系。

View 子类结构如图 3.1 所示。

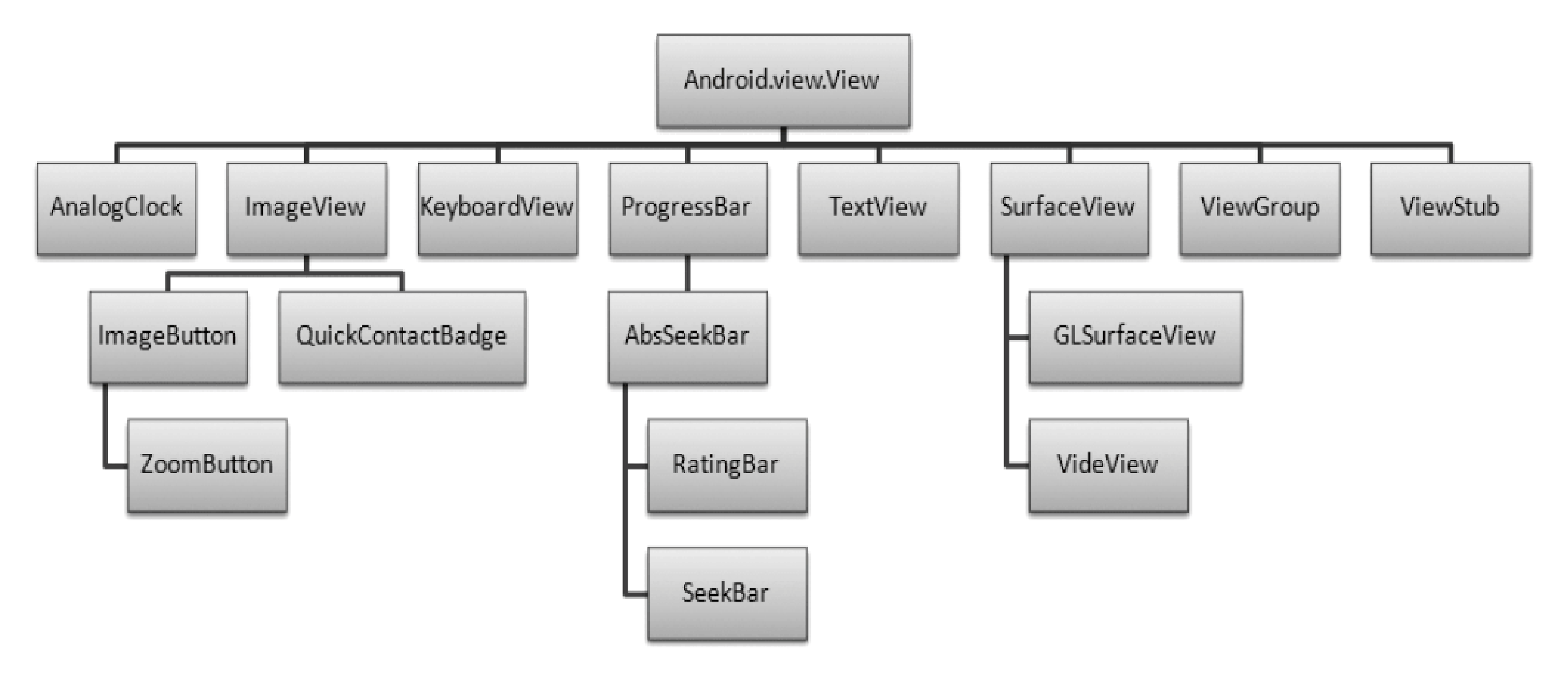


图 3.1　View 结构图

TextView 子类结构如图 3.2 所示。

ViewGroup 子类结构如图 3.3 所示。

FrameLayout 子类结构如图 3.4 所示。

Android 界面如图 3.5 所示。

通过学习本章内容,能够实现这个界面。下面介绍常用控件。

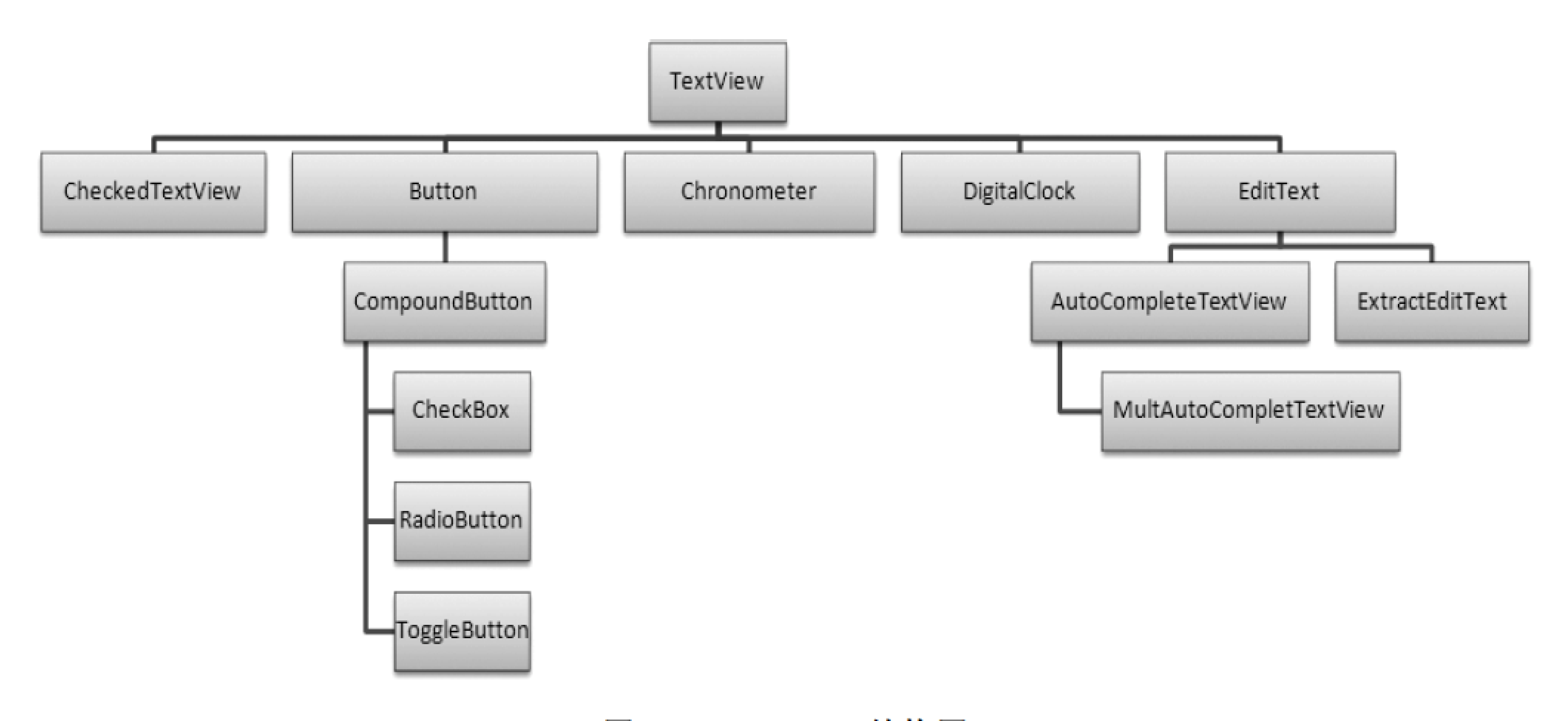


图 3.2　TextView 结构图

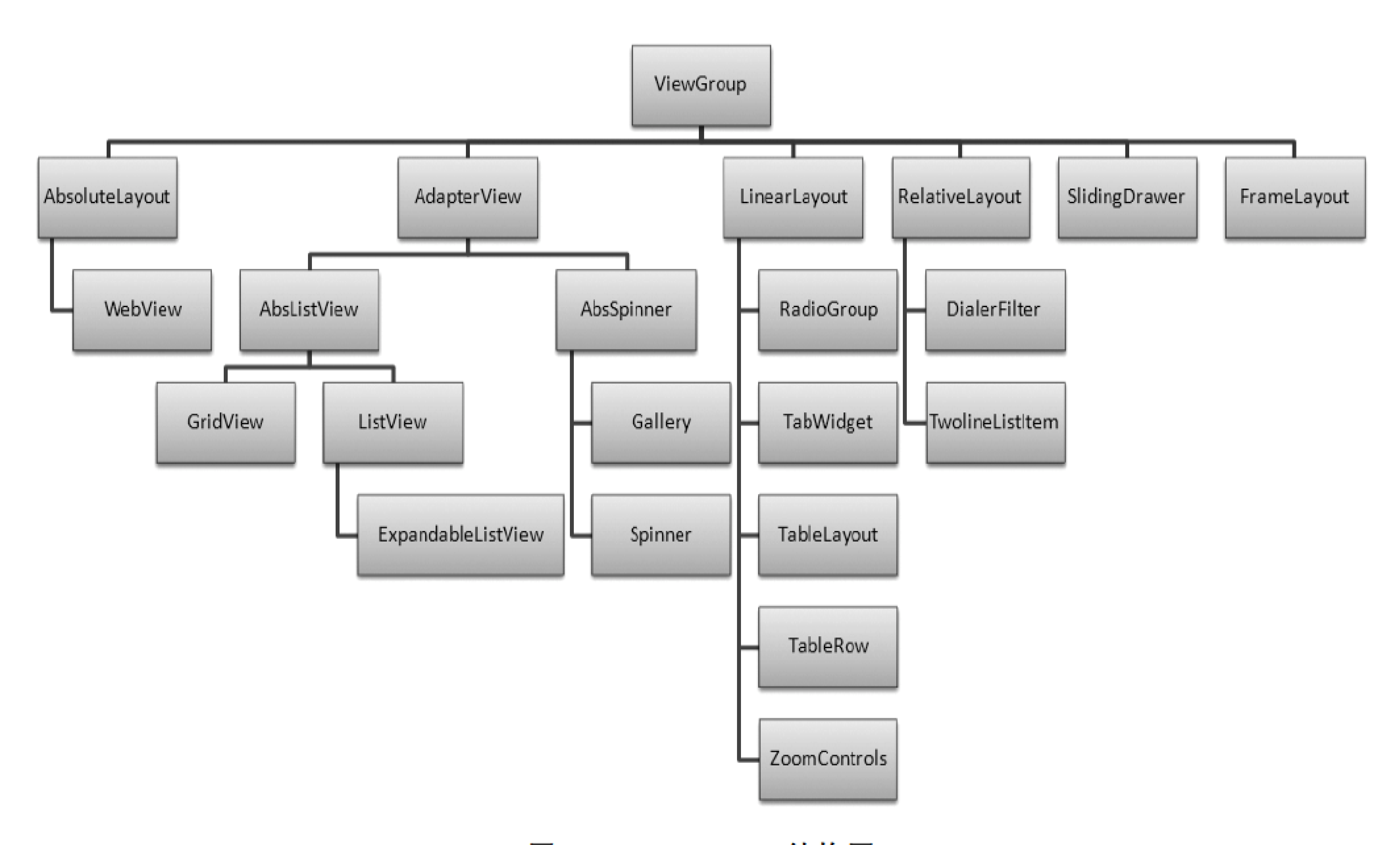


图 3.3　ViewGroup 结构图

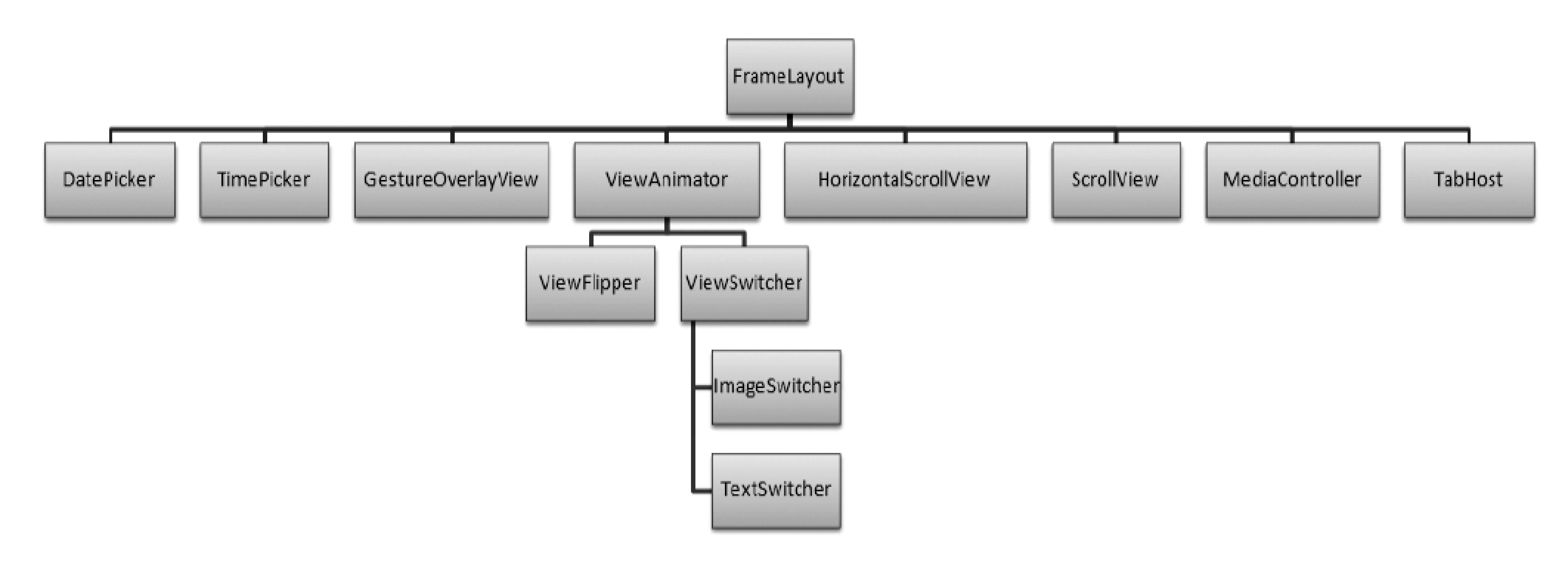


图 3.4　FrameLayout 结构图



图 3.5　注册界面

3.1　TextView

Android.widget.TextView 一般用于文本展示，继承自 Android.view.View，在 Android.widget 包中。常用子类有 Button、CheckedTextView、Chronometer、DigitalClock 和 EditText。

常用属性设置如表 3-1 所示。

限于篇幅，仅对基本用法举例说明，界面如图 3.6 所示。

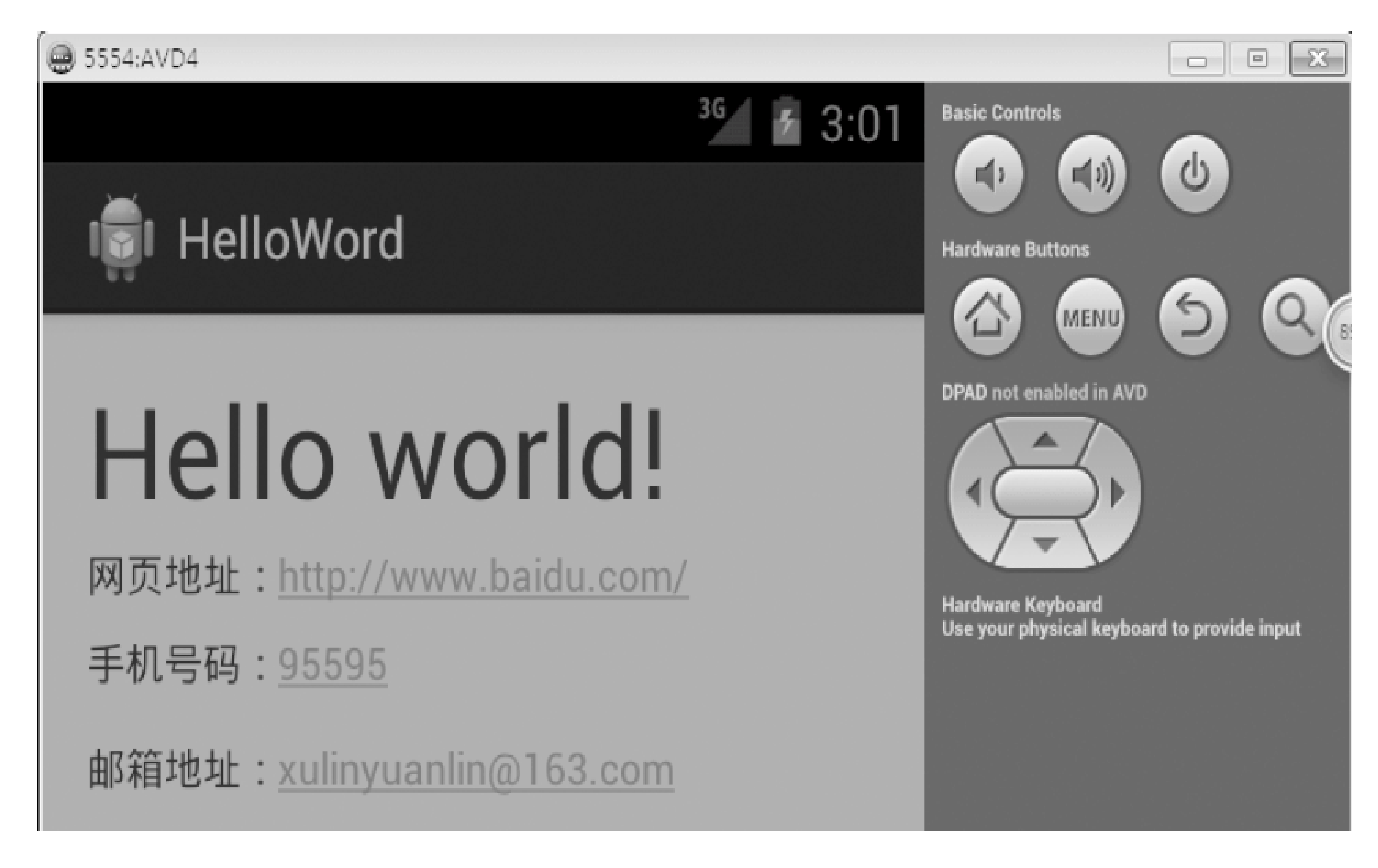


图 3.6　TextView

表 3-1　TextView 属性

属性名	关联方法	描述	取值
Android:autoLink	setAutoLinkMask(int)	控制是否自动查找并转换 URL、邮箱地址等链接为可单击链接。链接状态时，Web 情况可直接调用浏览器进行浏览；email 直接调用手机的 Email 软件；phone 转到拨打电话页面	默认值为“none”，禁止该属性。Web 网址，email 邮件，phone 电话，map 地图
Android:autoText	setKeyListener(KeyListener)	如果设置，这表示该 TextView 的文本输入法可以自动更正常见拼写错误	是布尔值，“true”或“false”
Android:bufferType	setText(CharSequence, TextView. BufferType)	决定 getText() 方法返回值的种类。注意，EditText 和 LogTextBox 总是返回 Editable	默认值为“normal”，可以返回任何字符序列，如果源文本是带格式的，也可以返回带格式的文本序列。Spannable，只能返回带格式的文本序列。Editable，只能返回带格式的或可编辑的文本序列
Android:capitalize	setKeyListener(KeyListener)	如果设置该属性，则指定该 TextView 的文本输入法可以自动改变用户输入的字母为大写	none，不进行首字母大写转换。sentences，每句的第一个单词首字母转为大写。words，每个单词的首字母大写。characters，所有字符大写。
Android:cursorVisible	setCursorVisible(boolean)	使光标可见(默认)或隐藏	布尔值，“true”或“false”
Android:digits	setKeyListener(KeyListener)	如果设置，则指定该 TextView 使用数字输入法，只接受指定的字符。False 时一定是字符串值，使用‘\\;’来转义字符。比如‘\\n’或用‘\\uxxxx’代 UNICODE 字符	默认值为 false。也可以是对包含该类型值的资源(形式为“@[package:]type:name”)或主题属性(形式为“? [package:][type:]name”)的参照
Android:drawableBottom	setCompoundDrawablesWithIntrinsicBounds(int, int, int, int)	在文本下方显示可绘制对象	可以是对其他资源的参照，形式为“@[+][package:] type: name”或“? [package:][type:]name”形式的主题属性 可能是颜色值，形式为“# rgb”、“# argb”、“# rrggbb”或“# aarrggbb”

续表

属 性 名	关 联 方 法	描 述	取 值
Android:drawableLeft	setCompoundDrawablesWithIntrinsicBounds(int,int,int,int)	在文本左侧显示可绘制对象	同上
Android:drawablePadding	setCompoundDrawablePadding(int)	文本和可绘制对象的间距	尺寸值,由浮点数后跟长度单位组成,比如"14.5sp"。可用单位有 px(像素)、dp(密度/设备无关像素)、sp(基于首选字体大小的缩放像素)、in(英寸)和 mm(毫米) 也可以是对包含该类型值的资源(形式为"@[package:]type:name")或主题属性(形式为"?[package:][type:]name")的参照
Android:drawableRight	setCompoundDrawablesWithIntrinsicBounds(int,int,int,int)	在文本右侧显示可绘制对象	同 Android:drawableBottom
Android:drawableTop	setCompoundDrawablesWithIntrinsicBounds(int,int,int,int)	在文本上方显示可绘制对象	同上
Android:editable		如果设置,则指定该 TextView 有输入法;如果没有特殊指定,其为文本输入状态。对于 TextView,该值默认为 false;对于 EditText,其默认值为 true	是布尔值,"true"或"false"。 也可以是对包含该类型值的资源(形式为"@[package:]type:name")或主题属性(形式为"?[package:][type:]name")的参照
Android:editorExtras	setInputExtras(int)	指向<input-extras>XML 资源,为输入法的特殊实现提供附加数据。只是当输入法启动时将资源放入 EditorInfo.extras 字段	对其他资源的参照,形式为"@[+][package:] type: name"或"?[package:][type:]name"形式的主题属性
Android:ellipsize	setEllipsize(TextUtils.TruncateAt)	该值为真时,如果文字长度超过视图宽度,文字不会在中途被截断,而是省略其中一部分。为了将文本作为一个整体显示在一行中,可能经常使用横向滚动条或者 singleLine 属性,现在也可以使用本属性,它还允许多行	none、start、middle、end、marquee

续表

属性名	关联方法	描述	取值
Android：ems	setEms(int)	使 TextView 精确匹配指定个数的字符宽度	是整数值，比如“100”。也可以是对包含该类型值的资源或主题属性的参照
Android：freezesText	setFreezesText(boolean)	如果设置，则文本视图会保持完整的文本内容，以及像当前光标位置这样的附近信息。默认该功能是关闭的，当文本视图的内容不是存储在内容提供者这样的永久资源中时，该功能很有用	是布尔值，“true”或“false”。也可以是对包含该类型值的资源或主题属性的参照
Android：gravity	setGravity(int)	指定当显示的文本比视图小时，横向和纵向的对齐方式	下列常量中的一个或多个（由'|'分割）。top：将对象放在其容器的顶部，不改变其大小。bottom：将对象放在其容器的底部，不改变其大小。left：将对象放在其容器的左侧，不改变其大小。right：将对象放在其容器的右侧，不改变其大小。center_vertical：将对象纵向居中，不改变其大小。fill_vertical：必要的时候增加对象的纵向大小，以完全充满其容器。center_horizontal：将对象横向居中，不改变其大小。fill_horizontal：必要的时候增加对象的横向大小，以完全充满其容器。center：将对象横纵居中，不改变其大小。fill：必要的时候增加对象的横纵向大小，以完全充满其容器。clip_vertical：附加选项，用于按照容器的边来剪切对象的顶部和/或底部的内容。剪切基于其纵向对齐设置：顶部对齐时，剪切底部；底部对齐时剪切顶部；除此之外剪切顶部和底部。clip_horizontal：附加选项，用于按照容器的边来剪切对象的左侧和/或右侧的内容。剪切基于其横向对齐设置：左侧对齐时，剪切右侧；右侧对齐时剪切左侧；除此之外剪切左侧和右侧

续表

属　性　名	关 联 方 法	描　　述	取　　值
Android:height	setHeight(int)	精确的设置 TextView 的高度。使用布局参数也可以达到相同效果	是尺寸值，由浮点数后跟长度单位组成，比如“14.5sp”。可用单位有 px(像素)、dp(密度/设备无关像素)、sp(基于首选字体大小的缩放像素)、in(英寸)和 mm(毫米)。也可以是对包含该类型值的资源或主题属性的参照
Android:hint	setHint(int)	当文本为空时显示的提示信息	是字符串值，使用‘\\;’来转义字符。比如‘\\n’或用‘\\uxxxx’代表 UNICODE 字符。也可以是对包含该类型值的资源或主题属性的参照
Android:imeActionId	setImeActionLabel(CharSequence,int)	当输入法连接到文本视图时，提供用于 EditorInfo.actionId 的值	整数值，比如“100”。也可以是对包含该类型值的资源或主题属性的参照
Android:imeActionLabel	setImeActionLabel(CharSequence,int)	当输入法连接到文本视图时，提供用于 EditorInfo.actionLabel 的值	是字符串值，使用‘\\;’来转义字符。比如‘\\n’或用‘\\uxxxx’代表 UNICODE 字符。也可以是对包含该类型值的资源或主题属性的参照
Android:imeOptions	setImeOptions(int)	附近特性，可用于启动输入法与编辑器的关联，以提高你应用程序的集成效果	是下列常量中的一个或多个(由'|'分割)。normal、actionUnspecified、actionNone、actionGo、actionSearch、actionSend、actionNext、actionDone、flagNoExtractU、flagNoAccessoryAction、flagNoEnterAction
Android:includeFontPadding	setIncludeFontPadding(boolean)	为上标和下标留出足够的空间，以取代字体上下标。默认为真	是布尔值，“true”或“false”。也可以是对包含该类型值的资源或主题属性的参照
Android:inputMethod	setKeyListener(KeyListener)	如果设置，即指定 TextView 使用指定的输入法(使用完全修饰类名)	是字符串值，使用‘\\;’来转义字符。比如‘\\n’或用‘\\uxxxx’代表 UNICODE 字符。也可以是对包含该类型值的资源或主题属性的参照

续表

属 性 名	关联方法	描 述	取 值
Android:inputType	setRawInputType(int)	文本域中可放置的数据类型,用于帮助输入法决定如何让用户输入文本	//文本类型,多为大写、小写和数字符号。text、textCapCharacters 字母大写、textCapWords 首字母大写、textCapSentences 仅第一个字母大写、textAutoCorrect 自动完成、textAutoComplete 自动完成、textMultiLine 多行输入、textImeMultiLine 输入法多行、textNoSuggestions 不提示、textUri 网址、textEmailAddress 电子邮件地址、textEmailSubject 邮件主题、textShortMessage 短讯、textLongMessage 长信息、textPersonName 人名、textPostalAddress 地址、textPassword 密码、textVisiblePassword 可见密码、textWebEditText 作为网页表单的文本、textFilter 文本筛选过滤、textPhonetic 拼音输入、//数值类型 number 数字、numberSigned 带符号数字格式、numberDecimal 带小数点的浮点格式、phone 拨号键盘、datetime 时间日期、date 日期键盘、time 时间键盘
Android:lineSpacingExtra	setLineSpacing(float,float)	文本行间距	是尺寸值,由浮点数后跟长度单位组成,比如"14.5sp"。可用单位有 px(像素)、dp(密度/设备无关像素)、sp(基于首选字体大小的缩放像素)、in(英寸)和 mm(毫米)。也可以是对包含该类型值的资源或主题属性的参照
Android:lineSpacingMultiplier	setLineSpacing(float,float)	用倍数指定的行间距	是浮点值,比如"1.2"。也可以是对包含该类型值的资源或主题属性的参照
Android:lines	setLines(int)	使 TextView 精确匹配指定行数的高度	是整数值,比如"100"。也可以是对包含该类型值的资源或主题属性的参照
Android:linksClickable	setLinksClickable(boolean)	如果设为假,即使指定的 autoLink 属性正确识别出了链接,单击也不会发生任何动作	是布尔值,"true"或"false"。也可以是对包含该类型值的资源或主题属性的参照

续表

属性名	关联方法	描述	取值
Android:marqueeRepeatLimit	setMarqueeRepeatLimit(int)	字幕动画的重复次数。仅应用于启动字母动画的 TextView	可能是整数值,比如"100"。也可以是对包含该类型值的资源或主题属性的参照。可能是常量 marquee_forever:表示该字幕动画将一直持续下去
Android:maxEms	setMaxEms(int)	使 TextView 的最大宽度为指定个数的字符宽度	是整数值,比如"100"。也可以是对包含该类型值的资源或主题属性的参照
Android:maxHeight	setMaxHeight(int)	设置 TextView 的最大高度	是尺寸值,由浮点数后跟长度单位组成,比如"14.5sp"。可用单位有 px(像素)、dp(密度/设备无关像素)、sp(基于首选字体大小的缩放像素)、in(英寸)和 mm(毫米)。也可以是对包含该类型值的资源或主题属性的参照
Android:maxLength	setFilters(InputFilter)	设置输入过滤器,限制输入的文本最大长度为指定值	是整数值,比如"100"。也可以是对包含该类型值的资源或主题属性的参照
Android:maxLines	setMaxLines(int)	使 TextView 的最大高度为指定行数的高度	同上
Android:maxWidth	setMaxWidth(int)	设置 TextView 的最大宽度	是尺寸值,由浮点数后跟长度单位组成,比如"14.5sp"。可用单位有 px(像素)、dp(密度/设备无关像素)、sp(基于首选字体大小的缩放像素)、in(英寸)和 mm(毫米)。也可以是对包含该类型值的资源或主题属性的参照
Android:minEms	setMinEms(int)	使 TextView 的最小宽度为指定个数的字符宽度	是整数值,比如"100"。也可以是对包含该类型值的资源或主题属性的参照
Android:minHeight	setMinHeight(int)	设置 TextView 的最小高度	是尺寸值,由浮点数后跟长度单位组成,比如"14.5sp"。可用单位有 px(像素)、dp(密度/设备无关像素)、sp(基于首选字体大小的缩放像素)、in(英寸)和 mm(毫米)。也可以是对包含该类型值的资源或主题属性的参照
Android:minLines	setMinLines(int)	使 TextView 的最小高度为指定行数的高度	是整数值,比如"100"。也可以是对包含该类型值的资源或主题属性的参照

续表

属性名	关联方法	描述	取值
Android:minWidth	setMinWidth(int)	设置 TextView 的最小宽度	是尺寸值，由浮点数后跟长度单位组成。比如“14.5sp”。可用单位有 px(像素)、dp(密度/设备无关像素)、sp(基于首选字体大小的缩放像素)、in(英寸)和 mm(毫米)。也可以是对包含该类型值的资源或主题属性的参照
Android:numeric	setKeyListener(KeyListener)	如果设置，指定 TextView 使用数值输入法，默认值为假	是下列常量中的一个或多个(由'|'分割)。integer:可以输入数值，signed:可以输入带符号的数值，decimal:可以输入带小数点的数值
Android:password	setTransformationMethod(TransformationMethod)	是否将文本域中的字符显示为代表密码的圆点	是布尔值，“true”或“false”。也可以是对包含该类型值的资源或主题属性的参照
Android:phoneNumber	setKeyListener(KeyListener)	如果设置，则指定 TextView 使用电话号码输入法，默认为假	是布尔值，“true”或“false”。也可以是对包含该类型值的资源或主题属性的参照
Android:privateImeOptions	setPrivateImeOptions(String)	附加的内容类型，用于描述文本视图支持的输入法的私有实现	是字符串值，使用‘\\;’来转义字符。比如‘\\n’或用‘\\uxxxx’代表 UNICODE 字符。也可以是对包含该类型值的资源或主题属性的参照
Android:scrollHorizontally	setHorizontallyScrolling(boolean)	是否允许文本比视图宽(允许横向滚动)	是布尔值，“true”或“false”。也可以是对包含该类型值的资源或主题属性的参照
Android:selectAllOnFocus	setSelectAllOnFocus(boolean)	如果文本可选，当该视图得到焦点时将文本全部选中，而不只将光标移动到开始或结尾处	是布尔值，“true”或“false”。也可以是对包含该类型值的资源或主题属性的参照
Android:shadowColor	setShadowLayer(float, float, float, int)	在文本后面显示指定颜色的阴影	是颜色值，形式为“#rgb”、“#argb”、“#rrggbb”或“#aarrggbb”。也可以是对包含该类型值的资源或主题属性的参照
Android:shadowDx	setShadowLayer(float, float, float, int)	阴影横向偏移量	是浮点值，比如“1.2”。也可以是对包含该类型值的资源或主题属性的参照
Android:shadowDy	setShadowLayer(float, float, float, int)	阴影纵向偏移量	同上

续表

属性名	关联方法	描述	取值
Android:shadowRadius	setShadowLayer(float,float,float,int)	设置阴影的范围	同上
Android:singleLine	setTransformationMethod(TransformationMethod)	限制文本显示与一行中,用横向滚动来代替多行显示。当你按下回车时,不是换行而是将文本向前移动。注意:对于可编辑文本视图,用 inputType 属性的 textMultiLine 标志来控制该行为(如果同时设置了 singleLine 和 inputType,inputType 的设置会覆盖 singleLine 的设置)	是布尔值,“true”或“false”。也可以是对包含该类型值的资源或主题属性的参照
Android:text	setText(CharSequence,TextView.BufferType)	用于显示的文本	是字符串值,使用‘\\;’来转义字符。比如‘\\n’或用‘\\uxxxx’代表 UNICODE 字符。也可以是对包含该类型值的资源或主题属性的参照
Android:textAppearance		基本的文字颜色、字体、大小和风格	是对其他资源的参照,形式为“@[+][package:]type:name”或“?[package:][type:]name”形式的主题属性
Android:textColor	setTextColor(ColorStateList)	文本颜色	是对其他资源的参照,形式为“@[+][package:]type:name”或“?[package:][type:]name”形式的主题属性。可能是颜色值,形式为“#rgb”、“#argb”、“#rrggbb”或“#aarrggbb”
Android:textColorHighlight	setHighlightColor(int)	选中文本的高亮部分的颜色	同上
Android:textColorHint	setHintTextColor(ColorStateList)	提示信息的文字颜色	同上
Android:textColorLink	setLinkTextColor(int)	超链接的文字颜色	同上

续表

属性名	关联方法	描述	取值
Android：textScaleX	setTextScaleX(float)	设置文本的横向缩放因子	是浮点值，比如“1.2”。也可以是对包含该类型值的资源或主题属性的参照
Android：textSize	setTextSize(float)	文字的大小。推荐以“sp(可缩放像素)”为单位来设置该值(比如15sp)	是尺寸值，由浮点数后跟长度单位组成，比如“14.5sp”。可用单位有px(像素)、dp(密度/设备无关像素)、sp(基于首选字体大小的缩放像素)、in(英寸)和mm(毫米)。也可以是对包含该类型值的资源或主题属性的参照
Android：textStyle	setTypeface(Typeface)	文字风格(粗体、斜体、粗斜体等)	是下列常量中的一个或多个(由'|'分割)。normal、bold、italic
Android：typeface	setTypeface(Typeface)	字体名称(楷体、宋体、仿宋、黑体等)	是下列常量之一。normal、sans、serif、monospace
Android：width	setWidth(int)	精确的设置TextView的宽度。使用布局参数也可以达到相同效果	是尺寸值，由浮点数后跟长度单位组成，比如“14.5sp”。可用单位有px(像素)、dp(密度/设备无关像素)、sp(基于首选字体大小的缩放像素)、in(英寸)和mm(毫米)。也可以是对包含该类型值的资源或主题属性的参照

该界面布局的 XML 代码如下：

```
<TextView
        Android:id="@+id/hellotitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:text="@string/hello_world"
        Android:textSize="40dp" />
    <TextView
        Android:id="@+id/webtitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignLeft="@+id/hellotitle"
        Android:layout_below="@+id/hellotitle"
        Android:layout_marginTop="18dp"
        Android:text="网页地址:" />
    <TextView
        Android:id="@+id/phonetitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignRight="@+id/webtitle"
        Android:layout_below="@+id/webtitle"
        Android:layout_marginTop="29dp"
        Android:text="手机号码:" />
    <TextView
        Android:id="@+id/emailtitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignLeft="@+id/phonetitle"
        Android:layout_below="@+id/phonetitle"
        Android:layout_marginTop="37dp"
        Android:text="邮箱地址:" />
    <TextView
        Android:id="@+id/weburl"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignBottom="@+id/webtitle"
        Android:layout_alignParentRight="true"
        Android:layout_marginRight="69dp"
        Android:text="http://www.baidu.com/"
        Android:autoLink="web"/>
    <TextView
        Android:id="@+id/phonenum"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignBaseline="@+id/phonetitle"
```

```
            Android:layout_alignBottom="@+id/phonetitle"
            Android:layout_alignLeft="@+id/weburl"
            Android:text="95595"
            Android:autoLink="phone"/>
        <TextView
            Android:id="@+id/emailurl"
            Android:layout_width="wrap_content"
            Android:layout_height="wrap_content"
            Android:layout_alignBaseline="@+id/emailtitle"
            Android:layout_alignBottom="@+id/emailtitle"
            Android:layout_alignLeft="@+id/phonenum"
            Android:text="xulinyuanlin@163.com"
            Android:autoLink="email"/>
```

该界面对应的 Activity 的 Java 代码如下：

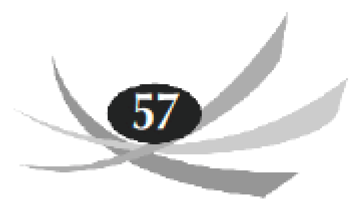
```
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.Menu;
import Android.widget.TextView;
public class TextViewActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_textview);
        TextView emailtitle= (TextView)super.findViewById(R.id.emailtitle);
        TextView emailurl= (TextView)super.findViewById(R.id.emailurl);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

TextView 可以动态修改显示的文字。只需在 onCreate 方法下面加入如下代码即可：

```
emailtitle.setText("Email 地址: ");                    //设置显示文字
```

3.2　EditText

Android.widget.EditText 输入框继承自 Android.widget.TextView，在 Android.widget 包中。常用子类有 AutoCompleteTextView、MultiAutoCompleteTextView 和 ExtractEditText。EditText 的属性继承自 TextView，内容大体相似。



下面举例说明基本用法。EditText 界面如图 3.7 所示。



图 3.7　EditText

该界面布局的 XML 代码如下：

```
<TextView
        Android:id="@+id/et_hltitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:text="@string/hello_world"
        Android:textSize="40dp" />
    <TextView
        Android:id="@+id/et_xmtitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignParentLeft="true"
        Android:layout_below="@+id/et_hltitle"
        Android:layout_marginTop="17dp"
        Android:text="姓名:" />
    <TextView
        Android:id="@+id/et_passtitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignLeft="@+id/et_xmtitle"
        Android:layout_below="@+id/et_xmtitle"
```

```
        Android:layout_marginTop="24dp"
        Android:text="密码:"/>
    <TextView
        Android:id="@+id/et_phonetitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignLeft="@+id/et_passtitle"
        Android:layout_below="@+id/et_passtitle"
        Android:layout_marginTop="26dp"
        Android:text="电话:"/>
    <TextView
        Android:id="@+id/et_emailtitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignLeft="@+id/et_phonetitle"
        Android:layout_below="@+id/et_phonetitle"
        Android:layout_marginTop="26dp"
        Android:text="邮箱:"/>
  <TextView
        Android:id="@+id/et_datetitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignLeft="@+id/et_emailtitle"
        Android:layout_below="@+id/et_emailtitle"
        Android:layout_marginTop="26dp"
        Android:text="日期:"/>
    <TextView
        Android:id="@+id/et_inttitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignLeft="@+id/et_datetitle"
        Android:layout_below="@+id/et_datetitle"
        Android:layout_marginTop="21dp"
        Android:text="整数:"/>
    <TextView
        Android:id="@+id/et_dectitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignLeft="@+id/et_inttitle"
        Android:layout_below="@+id/et_inttitle"
        Android:layout_marginTop="20dp"
        Android:text="小数:"/>
    <EditText
        Android:id="@+id/et_xm"
```

```
    Android:layout_width= "wrap_content"
    Android:layout_height= "wrap_content"
    Android:layout_alignBaseline= "@ + id/et_xmtitle"
    Android:layout_alignBottom= "@ + id/et_xmtitle"
    Android:layout_marginLeft= "38dp"
    Android:layout_toRightOf= "@ + id/et_xmtitle"
    Android:ems= "10"
    Android:inputType= "textPersonName" >
    < requestFocus />
< /EditText>
< EditText
    Android:id= "@ + id/et_pass"
    Android:layout_width= "wrap_content"
    Android:layout_height= "wrap_content"
    Android:layout_alignBaseline= "@ + id/et_passtitle"
    Android:layout_alignBottom= "@ + id/et_passtitle"
    Android:layout_alignLeft= "@ + id/et_xm"
    Android:ems= "10"
    Android:inputType= "textPassword" />
< EditText
    Android:id= "@ + id/et_phone"
    Android:layout_width= "wrap_content"
    Android:layout_height= "wrap_content"
    Android:layout_alignBaseline= "@ + id/et_phonetitle"
    Android:layout_alignBottom= "@ + id/et_phonetitle"
    Android:layout_alignLeft= "@ + id/et_pass"
    Android:ems= "10"
    Android:inputType= "phone"
    Android:phoneNumber= "true"/>
< EditText
    Android:id= "@ + id/et_date"
    Android:layout_width= "wrap_content"
    Android:layout_height= "wrap_content"
    Android:layout_alignBaseline= "@ + id/et_datetitle"
    Android:layout_alignBottom= "@ + id/et_datetitle"
    Android:layout_alignLeft= "@ + id/et_phone"
    Android:ems= "10"
    Android:inputType= "date" />
< EditText
    Android:id= "@ + id/et_int"
    Android:layout_width= "wrap_content"
    Android:layout_height= "wrap_content"
    Android:layout_alignBaseline= "@ + id/et_inttitle"
    Android:layout_alignBottom= "@ + id/et_inttitle"
```

```
        Android:layout_alignLeft= "@+id/et_email"
        Android:ems= "10"
        Android:inputType= "number" />
    <EditText
        Android:id= "@+id/et_dec"
        Android:layout_width= "wrap_content"
        Android:layout_height= "wrap_content"
        Android:layout_alignBaseline= "@+id/et_dectitle"
        Android:layout_alignBottom= "@+id/et_dectitle"
        Android:layout_alignLeft= "@+id/et_int"
        Android:ems= "10"
        Android:inputType= "numberDecimal" />
    <EditText
        Android:id= "@+id/et_email"
        Android:layout_width= "wrap_content"
        Android:layout_height= "wrap_content"
        Android:layout_alignBaseline= "@+id/et_emailtitle"
        Android:layout_alignBottom= "@+id/et_emailtitle"
        Android:layout_alignRight= "@+id/et_date"
        Android:ems= "10"
        Android:inputType= "textEmailAddress"/>
    <EditText
        Android:id= "@+id/et_decsign"
        Android:layout_width= "wrap_content"
        Android:layout_height= "wrap_content"
        Android:layout_alignBaseline= "@+id/et_decsigntitle"
        Android:layout_alignBottom= "@+id/et_decsigntitle"
        Android:layout_alignRight= "@+id/et_dec"
        Android:ems= "10"
        Android:inputType= "numberSigned" />
    <TextView
        Android:id= "@+id/et_decsigntitle"
        Android:layout_width= "wrap_content"
        Android:layout_height= "wrap_content"
        Android:layout_alignLeft= "@+id/et_dectitle"
        Android:layout_below= "@+id/et_dec"
        Android:layout_marginTop= "15dp"
        Android:text= "负数:" />
```

该界面对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.Menu;
import Android.widget.EditText;
```

```
public class EditTextActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edittext);
        EditText et_xm= (EditText)super.findViewById(R.id.et_xm);
        EditText et_pass= (EditText)super.findViewById(R.id.et_pass);
        EditText et_email= (EditText)super.findViewById(R.id.et_email);
        EditText et_phone= (EditText)super.findViewById(R.id.et_phone);
        EditText et_date= (EditText)super.findViewById(R.id.et_date);
        EditText et_int= (EditText)super.findViewById(R.id.et_int);
        EditText et_dec= (EditText)super.findViewById(R.id.et_dec);
        EditText et_decsign= (EditText)super.findViewById(R.id.et_decsign);
        //获取输入的信息
        String strName= et_xm.getText().toString();
        String strPass= et_pass.getText().toString();
        String strEmail= et_email.getText().toString();
        String strPhone= et_phone.getText().toString();
        String strDate= et_date.getText().toString();
        String strint= et_int.getText().toString();
        String strDec= et_dec.getText().toString();
        String strDecsign= et_decsign.getText().toString();
    }
}
```

3.3 AutoCompleteTextView

Android.widget.AutoCompleteTextView 带提示的输入框继承自 Android.widget.EditText，在 Android.widget 包中。子类是 MultiAutoCompleteTextView。

AutoCompleteTextView 和 MultiAutoCompleteTextView 都是自动提示，一个是单选，一个是多选。

AutoCompleteTextView 常用属性如表 3-2 所示。

表 3-2 AutoCompleteTextView 属性

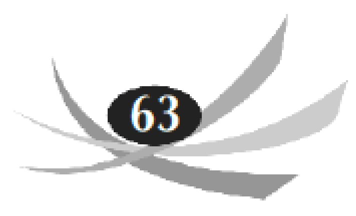
属 性	关联方法	说明	取 值
Android:completionHint	setCompletionHint(CharSequence)	在下拉列表中显示的提示	是字符串值，使用'\\;'来转义字符。比如'\\n'或用'\\uxxxx'代表 UNICODE 字符。也可以是对包含该类型值的资源或主题属性的参照
Android:completionHintView		在下拉列表中显示的提示关联的 view	是对其他资源的参照，形式为"@[+][package:]type:name"或"?[package:][type:]name"形式的主题属性

续表

属　性	关联方法	说明	取　值
Android：completionThreshold	setThreshold(int)	输入几个字符时提示	是整数值，比如"100"。也可以是对包含该类型值的资源或主题属性的参照
Android：dropDownAnchor	setDropDownAnchor(int)	针对锚自动完成下拉	是对其他资源的参照，形式为"@[+][package:]type:name"或"?[package:][type:]name"形式的主题属性
Android：dropDownHeight	setDropDownHeight(int)	指定下拉基础高度	是尺寸值，由浮点数后跟长度单位组成，比如"14.5sp"。可用单位有px(像素)、dp(密度/设备无关像素)、sp(基于首选字体大小的缩放像素)、in(英寸)和mm(毫米)。也可以是对包含该类型值的资源或主题属性的参照。可能是下列常量之一：fill_parent、match_parent、wrap_content
Android：dropDownHorizontal-Offset		水平像素偏移量	是尺寸值，由浮点数后跟长度单位组成，比如"14.5sp"。可用单位有px(像素)、dp(密度/设备无关像素)、sp(基于首选字体大小的缩放像素)、in(英寸)和mm(毫米)。也可以是对包含该类型值的资源或主题属性的参照
Android：dropDownSelector		在下拉列表中选择	可能是对其他资源的参照，形式为"@[+][package:]type:name"或"?[package:][type:]name"形式的主题属性。可能是颜色值，形式为"#rgb"、"#argb"、"#rrggbb"或"#aarrggbb"
Android：dropDownVertical-Offset		垂直像素偏移量	是尺寸值，由浮点数后跟长度单位组成，比如"14.5sp"。可用单位有px(像素)、dp(密度/设备无关像素)、sp(基于首选字体大小的缩放像素)、in(英寸)和mm(毫米)。也可以是对包含该类型值的资源或主题属性的参照
Android：dropDownWidth	setDropDownWidth(int)	指定下拉基础宽度	是尺寸值，由浮点数后跟长度单位组成，比如"14.5sp"。可用单位有px(像素)、dp(密度/设备无关像素)、sp(基于首选字体大小的缩放像素)、in(英寸)和mm(毫米)。也可以是对包含该类型值的资源或主题属性的参照。可能是下列常量之一：fill_parent、match_parent、wrap_content
Android：popupBackground	setDropDownBackgroundResource(int)		

基本用法：

(1) AutoCompleteTextView可以看作是一个带自动提示的EditText，当输入字符时，会出现提示窗口，选择即可。



在 layout 中定义一个 AutoCompleteTextView，然后在 Activity 中设置数据源。数据源 ArrayAdapter 的构造方法的 3 个参数为上下文的 Context、每行的 textView 布局和数据源。

Activity 的 Java 代码如下：

```
this.autoCompleteTextView= (AutoCompleteTextView) super.findViewById(R.id.autoCompleteTextView);
ArrayAdapter< String> arrayAdapter= new ArrayAdapter< String> (this, R.layout.
arrayadapte_textview, CITY_NAMES);
this.autoCompleteTextView.setAdapter(arrayAdapter);
```

（2）MultiAutoCompleteTextView 也是带有提示的输入框。与 AutoCompleteTextView 的区别在于可以连续提示，选择一个提示项后会自动添加一个分隔符，在输入时继续提示。AutoCompleteTextView 则属于单选模式的，使用时需要设置分隔符类 CommaTokenizer。其他与 AutoCompleteTextView 一样。

举例说明，界面如图 3.8 所示。

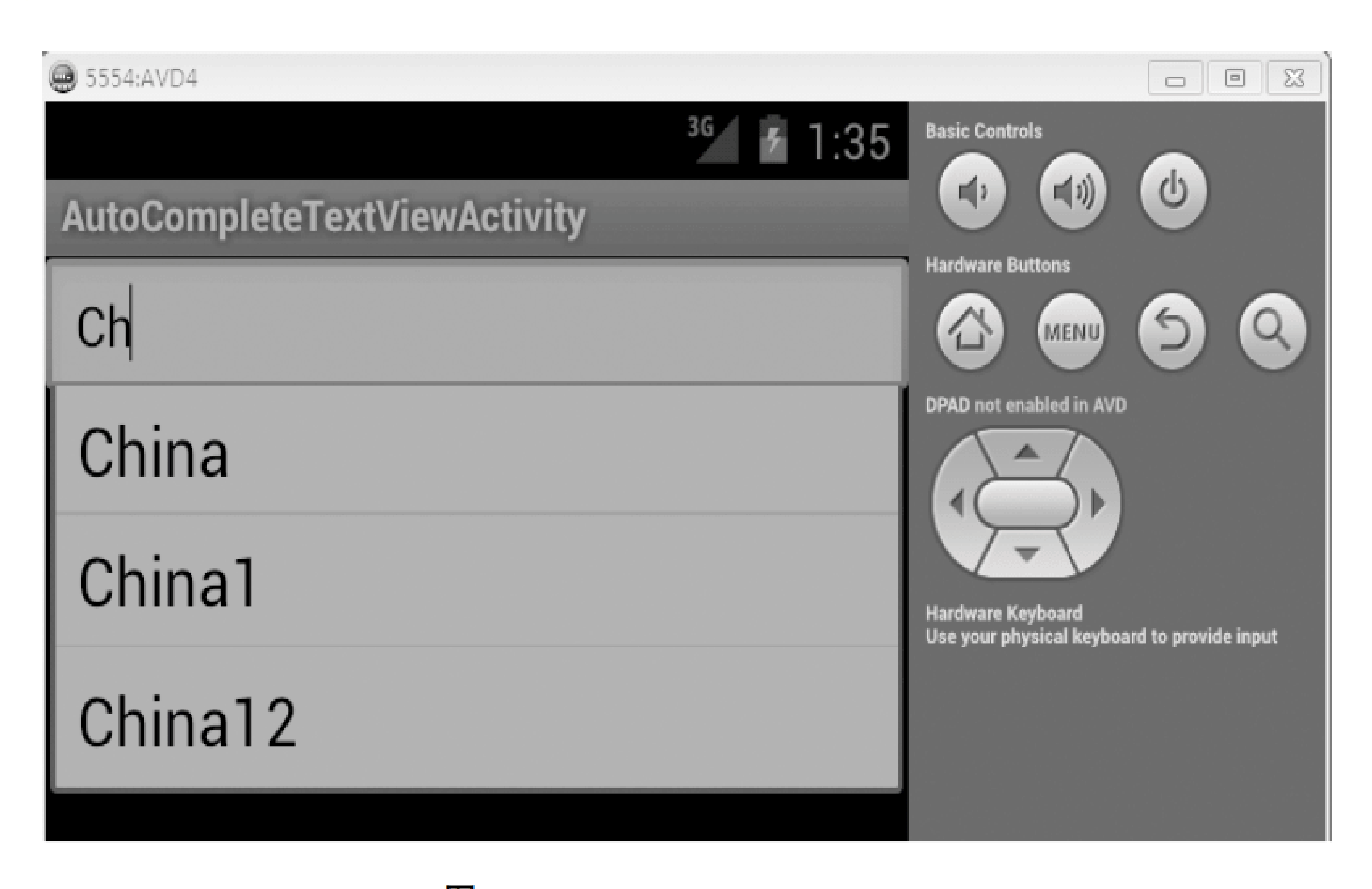


图 3.8　AutoCompleteTextView

该界面布局的 XML 代码如下：

```
< AutoCompleteTextView Android:id= "@+ id/auto_complete"
        Android:layout_width= "fill_parent"
        Android:layout_height= "wrap_content"/>
```

该界面对应的 Activity 如下(Java 代码)：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.widget.ArrayAdapter;
import Android.widget.AutoCompleteTextView;
public class AutoCompleteTextViewActivity extends Activity {
        @Override
        protected void onCreate(Bundle savedInstanceState) {
```

```
                super.onCreate(savedInstanceState);
                setContentView(R.layout.autocomplete);
                setTitle("AutoCompleteTextViewActivity");
                ArrayAdapter<String> adapter= new ArrayAdapter<String>(this,
                        Android.R.layout.simple_dropdown_item_1line, COUNTRIES);
                AutoCompleteTextView textView= (AutoCompleteTextView) findViewById
                (R.id.auto_complete);
                textView.setAdapter(adapter);
            }
        static final String[] COUNTRIES= new String[] {
            "China","Russia", "Germany",
            "Ukraine", "Belarus", "USA","China1","China12", "Germany",
            "Russia2", "Belarus", "USA"
            };
    }
```

3.4 Button

Android.widget.Button 是最常用的按钮，继承自 Android.widget.TextView，在 Android.widget 包中。常用子类有 CheckBox、RadioButton 和 ToggleButton。

属性与 Android.widget.TextView 相似。

用法如下：super.findViewById(id)得到在 layout 中声明的 Button 的引用，添加监听 setOnClickListener(View.OnClickListener)。在 View.OnClickListener 监听器中使用 v.equals(View)方法判断哪个按钮被按下，分别进行处理。

举例说明，界面如图 3.9 所示。

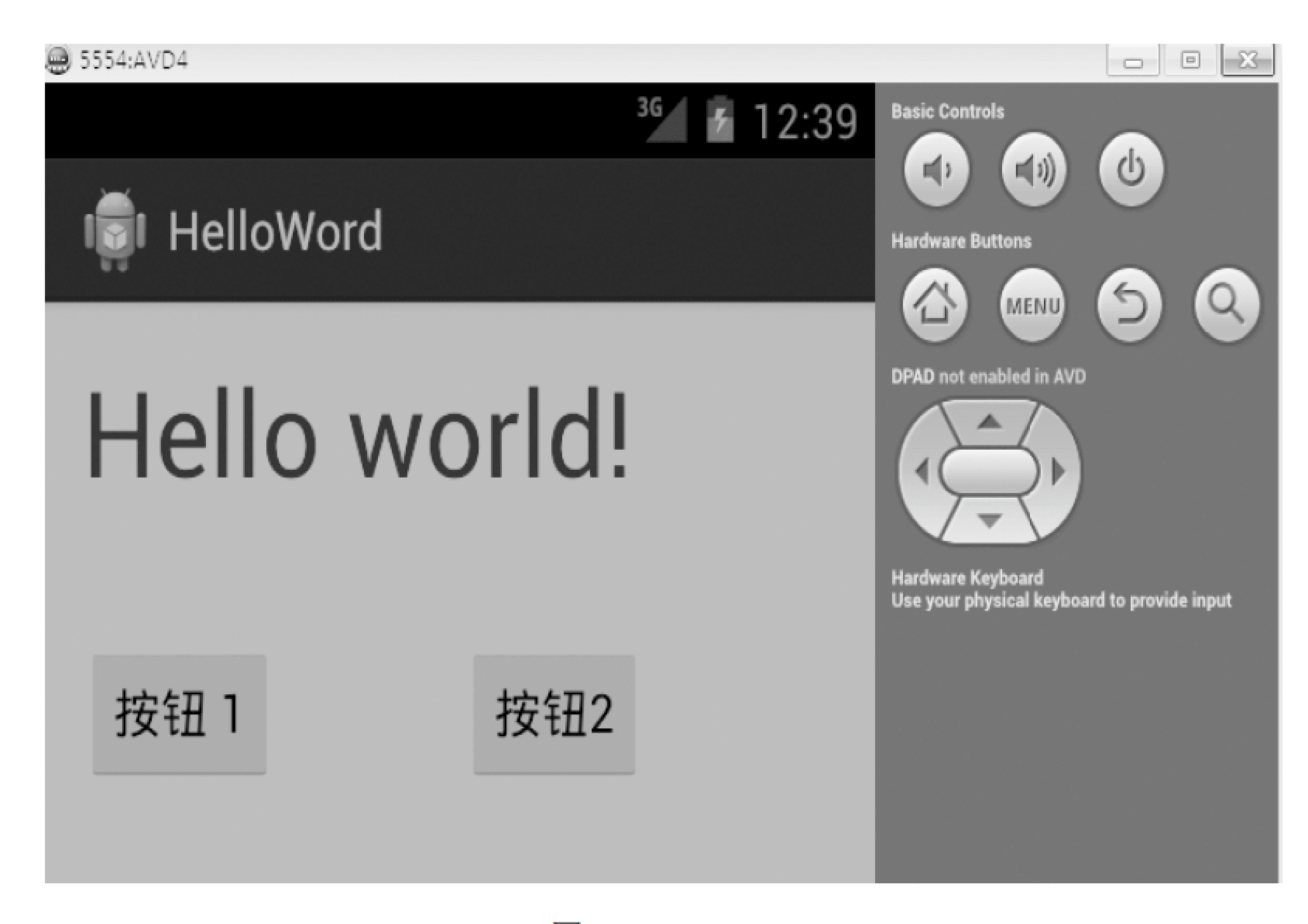


图 3.9 Button

该界面布局的 XML 代码如下：

```
<TextView
        Android:id="@+id/bt_hltitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:text="@string/hello_world"
        Android:textSize="40dp" />
    <Button
        Android:id="@+id/bt_ok"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignLeft="@+id/bt_hltitle"
        Android:layout_below="@+id/bt_hltitle"
        Android:layout_marginTop="44dp"
        Android:text="按钮 1" />
    <Button
        Android:id="@+id/bt_cancel"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignBottom="@+id/bt_ok"
        Android:layout_alignParentRight="true"
        Android:layout_marginRight="72dp"
        Android:text="按钮 2" />
```

该界面对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.Menu;
import Android.view.View;
import Android.view.View.OnClickListener;
import Android.widget.Button;
public class ButtonActivity extends Activity {
    Button button1;
    Button button2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_button);
        button1= (Button)super.findViewById(R.id.bt_ok);
        button2= (Button)super.findViewById(R.id.bt_cancel);
        button1.setOnClickListener(new OnClickListener() {
            public void onClick(View v)
            {
```

```
                setTitle("我是按钮 1");
            }
        });
        button2.setOnClickListener(new OnClickListener(){
            public void onClick(View v)
            {
                setTitle("我是按钮 2");
            }
        });
    }
}
```

3.5　CheckBox

Android.widget.CheckBox 是复选按钮，继承自 Android.widget.CompoundButton，在 Android.widget 包中。

属性与 Android.widget.TextView 相似。

用法如下：使用 isChecked()检查是否被选中。需要添加监听 setOnCheckedChangeListener(CompoundButton.OnCheckedChangeListener)。

举例说明，界面如图 3.10 所示。

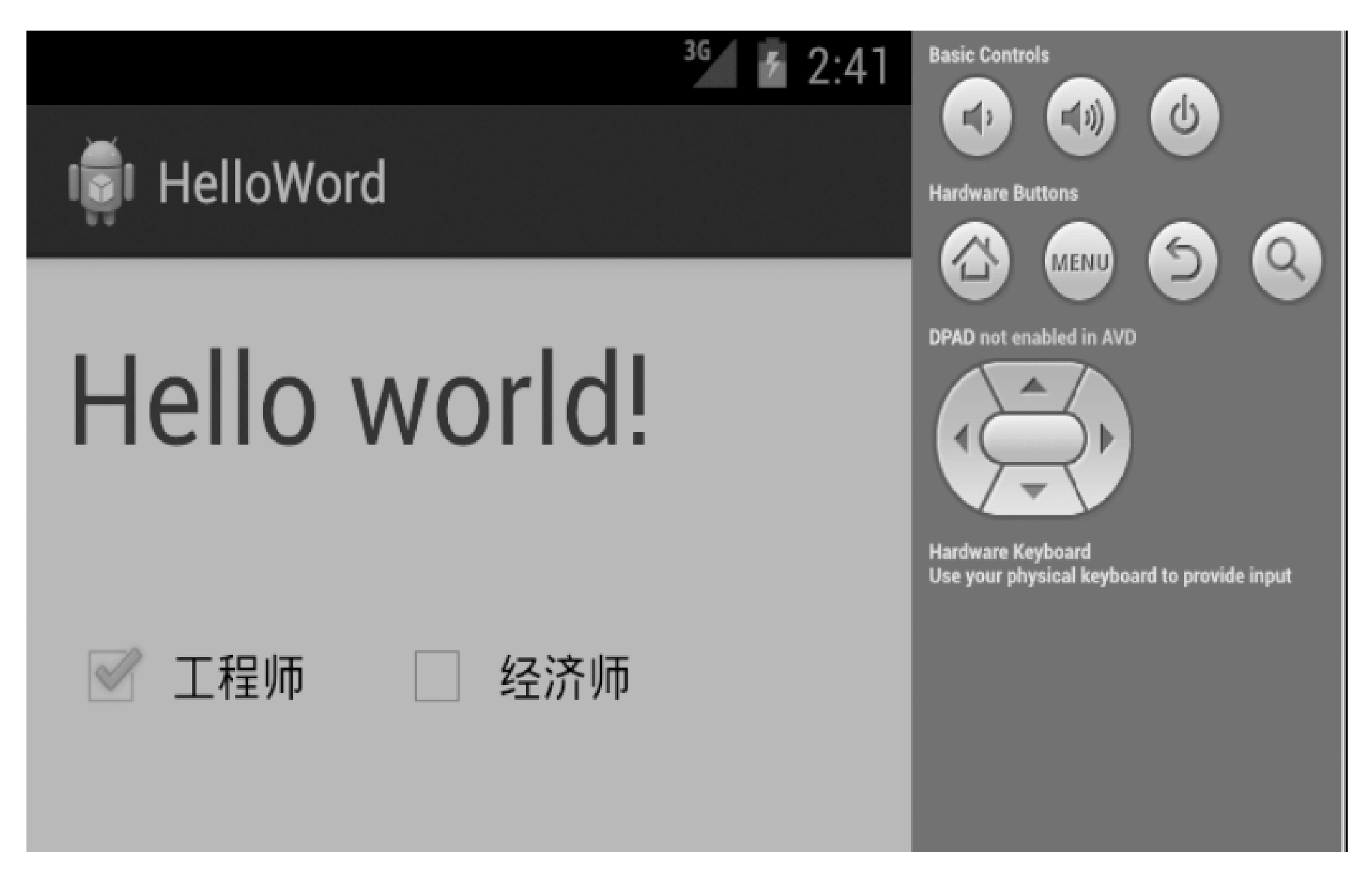


图 3.10　CheckBox

该界面布局的 XML 代码如下：

```
<TextView
        Android:id="@+id/cb_hltitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:text="@string/hello_world"
```

```
        Android:textSize= "40dp" />
    <CheckBox
        Android:id= "@+ id/cb_gcs"
        Android:layout_width= "wrap_content"
        Android:layout_height= "wrap_content"
        Android:layout_alignLeft= "@+ id/cb_hltitle"
        Android:layout_below= "@+ id/cb_hltitle"
        Android:layout_marginTop= "42dp"
        Android:text= "工程师" Android:checked= "true"/>
    <CheckBox
        Android:id= "@+ id/cb_jjs"
        Android:layout_width= "wrap_content"
        Android:layout_height= "wrap_content"
        Android:layout_alignBaseline= "@+ id/cb_gcs"
        Android:layout_alignBottom= "@+ id/cb_gcs"
        Android:layout_marginLeft= "31dp"
        Android:layout_toRightOf= "@+ id/cb_gcs"
        Android:text= "经济师" />
```

该界面对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.Menu;
import Android.widget.CheckBox;
import Android.widget.CompoundButton;
import Android.widget.CompoundButton.OnCheckedChangeListener;
public class CheckBoxActivity extends Activity {
    CheckBox button1;
    CheckBox button2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_checkbox);
        button1= (CheckBox)super.findViewById(R.id.cb_gcs);
        button2= (CheckBox)super.findViewById(R.id.cb_jjs);
        button1.setOnCheckedChangeListener(new OnCheckedChangeListener(){
          @Override
            public void onCheckedChanged(CompoundButton arg0, boolean arg1) {
                String r= "";
                if(button1.isChecked()) {
                    r= r+ ","+ button1.getText();
                }
                if(button2.isChecked()) {
                    r= r+ ","+ button2.getText();
                }
```

```
                setTitle("Checked: "+ r);
            }
        });
        button2.setOnCheckedChangeListener(new OnCheckedChangeListener(){
          @Override
            public void onCheckedChanged(CompoundButton arg0, boolean arg1) {
                String r= "";
                if(button1.isChecked()) {
                    r= r+ ","+ button1.getText();
                }
                if(button2.isChecked()) {
                    r= r+ ","+ button2.getText();
                }
                setTitle("Checked: "+ r);
            }
        });
    }
}
```

3.6 RadioButton

Android.widget.RadioButton 是单选按钮，继承自 Android.widget.CompoundButton，在 Android.widget 包中。

属性与 Android.widget.TextView 相似。

用法如下：单选按钮要声明在 RadioGroup 中，RadioGroup 是流式布局 Android.widget.LinearLayout 的子类。需要向 RadioGroup 添加状态更改的监听 setOnCheckedChangeListener(RadioGroup.OnCheckedChangeListener)。监听器类型和 CheckBox 是不一样的。

举例说明，界面如图 3.11 所示。

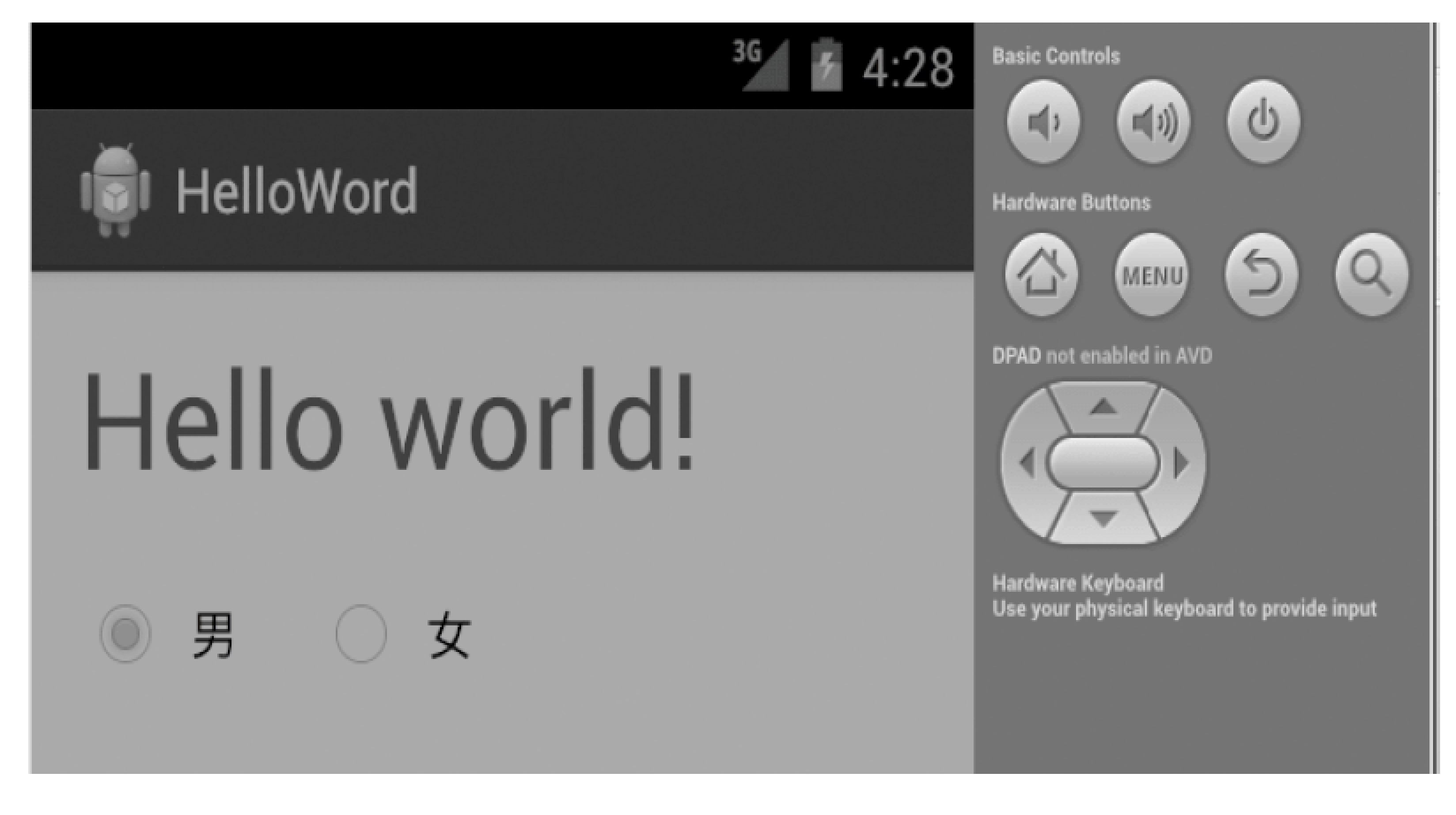


图 3.11 RadioButton

该界面布局的 XML 代码如下：

```
<TextView
        Android:id="@+id/rb_hltitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:text="@string/hello_world"
        Android:textSize="40dp" />
    <RadioButton
        Android:id="@+id/rb_man"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignLeft="@+id/rb_hltitle"
        Android:layout_below="@+id/rb_hltitle"
        Android:layout_marginTop="18dp"
        Android:text="男" Android:checked="true"/>
    <RadioButton
        Android:id="@+id/rb_women"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignBaseline="@+id/rb_man"
        Android:layout_alignBottom="@+id/rb_man"
        Android:layout_marginLeft="26dp"
        Android:layout_toRightOf="@+id/rb_man"
        Android:text="女" />
```

该界面对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.Menu;
import Android.widget.CompoundButton;
import Android.widget.CompoundButton.OnCheckedChangeListener;
import Android.widget.RadioButton;
public class RadioButtonActivity extends Activity {
    RadioButton button1;
    RadioButton button2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_radiobutton);
        button1= (RadioButton)super.findViewById(R.id.rb_man);
        button2= (RadioButton)super.findViewById(R.id.rb_women);
        button1.setOnCheckedChangeListener(new OnCheckedChangeListener(){
```

```
            @Override
            public void onCheckedChanged(CompoundButton arg0, boolean arg1) {
                setTitle(button1.getText());
            }
        });
        button2.setOnCheckedChangeListener(new OnCheckedChangeListener(){
            @Override
            public void onCheckedChanged(CompoundButton arg0, boolean arg1) {
                setTitle(button2.getText());
            }
        });
    }
}
```

3.7 ToggleButton

Android. widget. ToggleButton 是开关形式的按钮，继承自 Android. widget. CompoundButton，在 Android. widget 包中。属性设置如表 3-3 所示。

表 3-3 ToggleButton 属性设置

属　性	说　明
Android:textOn=""	选择状态文字
Android:textOff=""	未选状态文字

举例说明，界面如图 3.12 和图 3.13 所示。

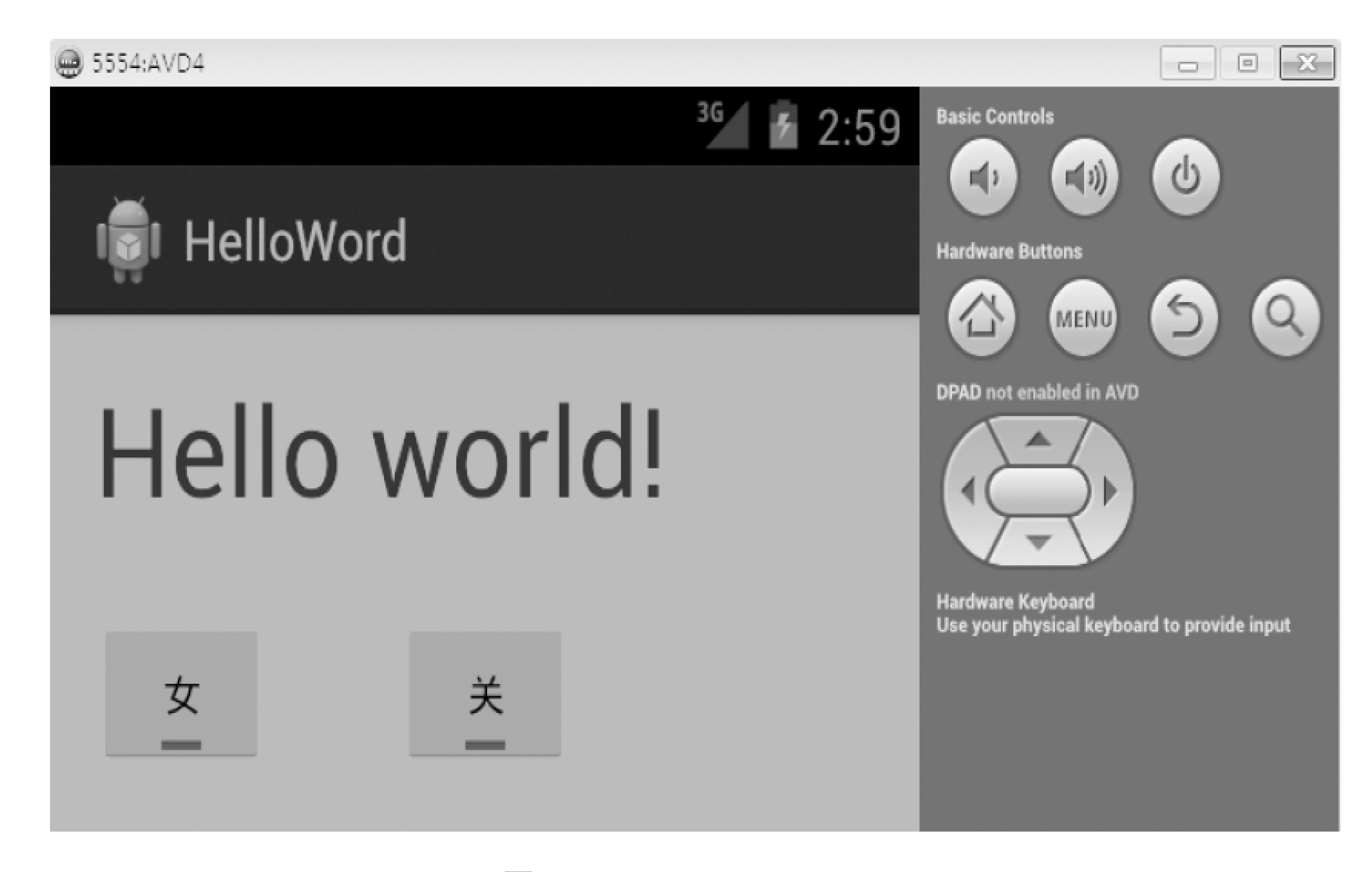


图 3.12 ToggleButton(1)

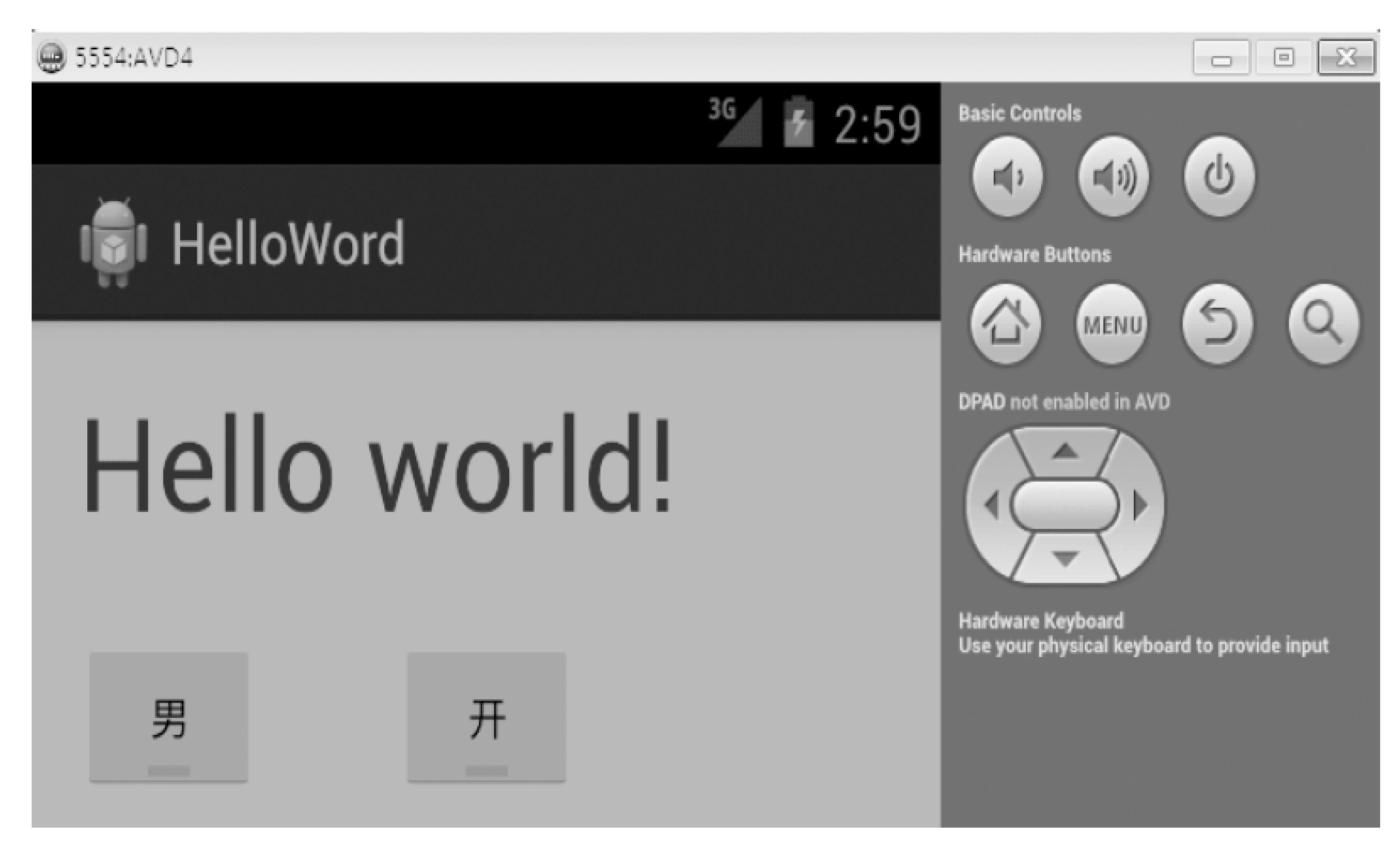


图 3.13　ToggleButton(2)

该界面布局的 XML 代码如下：

```
< TextView
        Android:id= "@ + id/tb_hltitle"
        Android:layout_width= "wrap_content"
        Android:layout_height= "wrap_content"
        Android:text= "@ string/hello_world"
        Android:textSize= "40dp" />
    < ToggleButton
        Android:id= "@ + id/tg_sex"
        Android:layout_width= "wrap_content"
        Android:layout_height= "wrap_content"
        Android:layout_alignLeft= "@ + id/tb_hltitle"
        Android:layout_below= "@ + id/tb_hltitle"
        Android:layout_marginTop= "28dp"
        Android:textOn= "男 "
        Android:textOff= "女 " />
    < ToggleButton
        Android:id= "@ + id/tg_onoff"
        Android:layout_width= "wrap_content"
        Android:layout_height= "wrap_content"
        Android:layout_alignBottom= "@ + id/tg_sex"
        Android:layout_centerHorizontal= "true"
        Android:textOn= "开 "
        Android:textOff= "关 "  />
```

该界面对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.widget.ToggleButton;
public class ToggleButtonActivity extends Activity {
```

```
        ToggleButton toggleButton1;
        ToggleButton toggleButton2;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_togglebutton);
            toggleButton1= (ToggleButton) super.findViewById(R.id.tg_sex);
            toggleButton2= (ToggleButton) super.findViewById(R.id.tg_onoff);
        }
    }
```

3.8 ImageView

Android.widget.ImageView 是图片控件，继承自 Android.view.View，在 Android.widget 包中。

用法如下：

(1) src 设置图片路径，可引用 drawable 的图片。Layout 代码如下：

```
<ImageView Android:layout_width= "wrap_content"
    Android:layout_height= "wrap_content"
                    Android:src= "@drawable/tool"/>
```

(2) 动态声明 ImageView，设置 src。

举例说明，界面如图 3.14 所示。

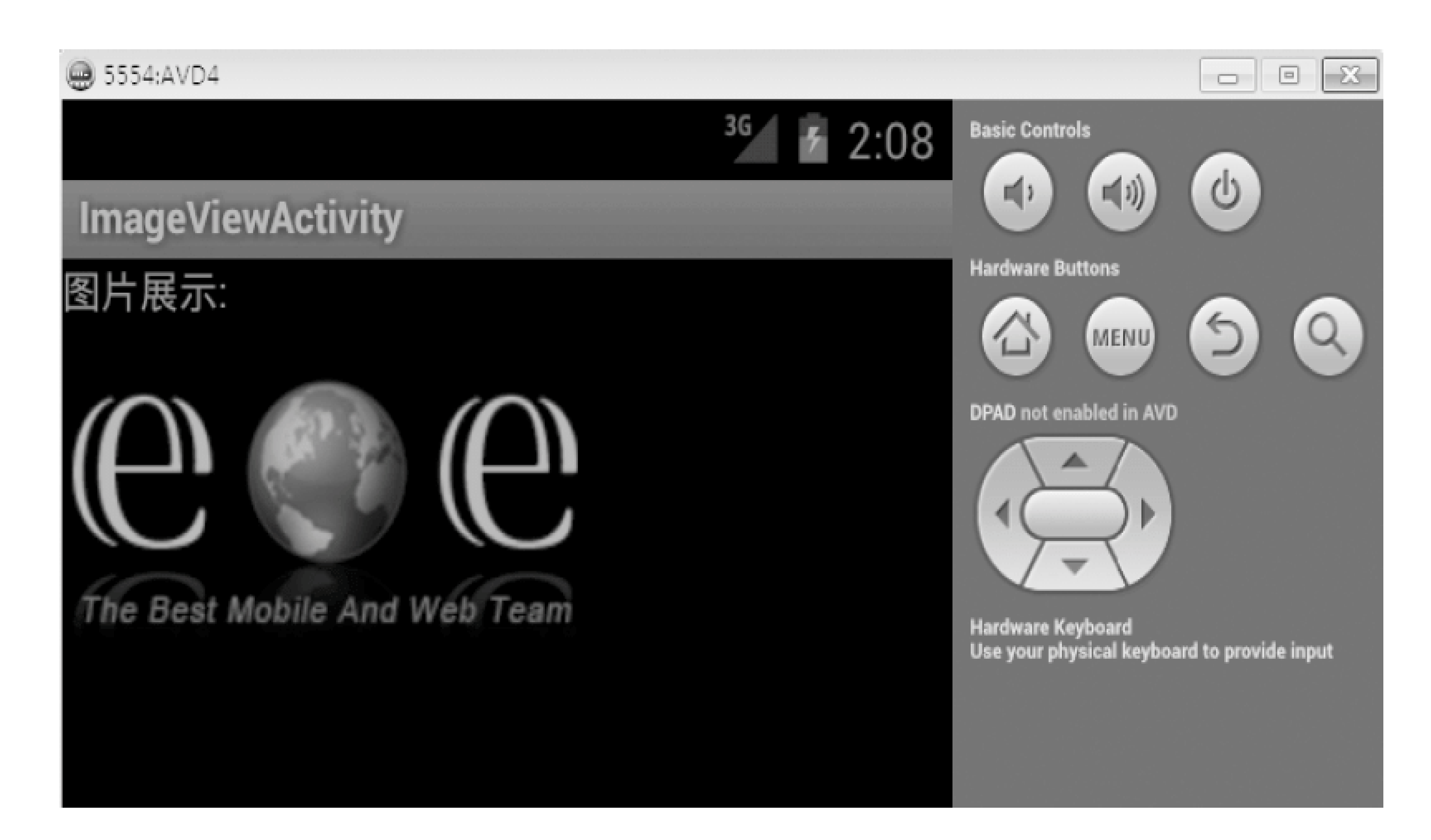
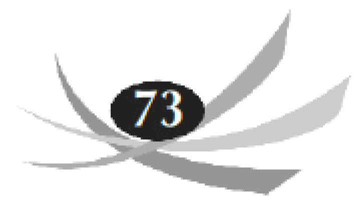


图 3.14 ImageView

该界面布局的 XML 代码如下：

```
<TextView
        Android:layout_width= "wrap_content"
        Android:layout_height= "wrap_content"
        Android:text= "图片展示:" />
```




```
< ImageView
   Android:id= "@ + id/imagebutton"
   Android:src= "@ drawable/view"
   Android:layout_width= "wrap_content"
   Android:layout_height= "wrap_content"/>
```

该界面对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.widget.ImageView;
public class ImageViewActivity extends Activity {
      @ Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setTitle("ImageViewActivity");
            setContentView(R.layout.image_view);
ImageView img= (ImageView)super.findViewById(R.id.imagebutton);
      }
}
```

3.9　ImageButton

Android.widget.ImageButton 是图片控件，继承自 Android.widget.ImageView，在 Android.widget 包中。

用法如下：

(1) src 设置图片路径，可引用 drawable 的图片。Layout 代码如下：

```
< ImageButton Android:layout_width= "wrap_content"
Android:layout_height= "wrap_content" Android:src= "@ drawable/but_01"/>
```

(2) 动态声明 ImageView，设置 src。

举例说明，界面如图 3.15 所示。

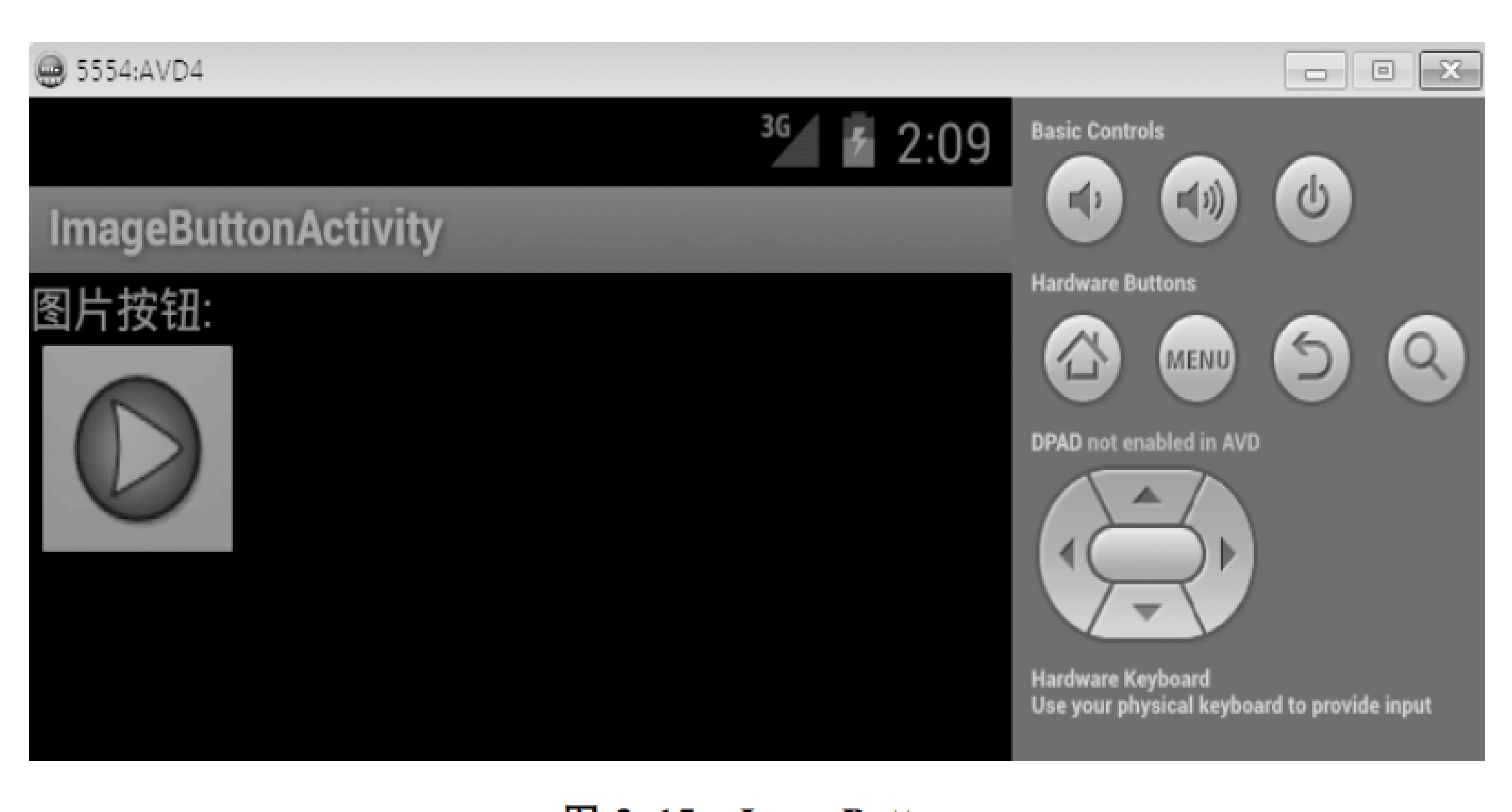


图 3.15　ImageButton

该界面布局的XML代码如下：

```
<TextView
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:text="图片按钮:"/>
<ImageButton Android:id="@+id/imagebutton"
  Android:src="@drawable/image_button"
  Android:layout_width="wrap_content"
  Android:layout_height="wrap_content"/>
```

该界面对应的Activity的Java代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.widget.ImageButton;
public class ImageButtonActivity extends Activity {
ImageButton button;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setTitle("ImageButtonActivity");
        setContentView(R.layout.image_button);
        button=(ImageButton)super.findViewById(R.id.image_button_button);
    }
}
```

3.10 ImageSwitcher 和 Gallery

Android.widget.ImageSwitcher是图片控件，继承自Android.widget.ViewSwitcher(ViewGroup)，在Android.widget包中。

用法如下：

(1) ImageSwithcer用来控制图片，使用方法setInAnimation(Animation)、setOutAnimation(Animation)设置动画。

(2) Gallery用来控制图片显示区域下面的图片索引列表。ImageAdapter继承自BaseAdapter，设置Gallery的适配器。

(3) 在layout中添加ImageSwitcher和Gallery。定义Activity、implements接口OnItemSelectedListener、ViewFactory。在Gallery的onCreate方法中定义需要显示的图片路径列表，设置Gallery的Adapter。onItemSelected事件触发时，设置对应的图片。

举例说明，界面如图3.16所示。

该界面布局的XML代码如下：

```
<ImageSwitcher
      Android:id="@+id/switcher"
```



图 3.16　ImageSwitcher 和 Gallery

```
        Android:layout_width="fill_parent"
        Android:layout_height="fill_parent"
        Android:layout_alignParentTop="true"
        Android:layout_alignParentLeft="true" />
<Gallery Android:id="@+id/gallery"
Android:background="#55000000"
        Android:layout_width="fill_parent"
        Android:layout_height="60dp"
        Android:layout_alignParentBottom="true"
        Android:layout_alignParentLeft="true"
        Android:gravity="center_vertical"
        Android:spacing="16dp" />
```

该界面对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.content.Context;
import Android.os.Bundle;
import Android.view.View;
import Android.view.ViewGroup;
import Android.view.Window;
import Android.view.animation.AnimationUtils;
import Android.widget.AdapterView;
import Android.widget.BaseAdapter;
```

```
import Android.widget.Gallery;
import Android.widget.ImageSwitcher;
import Android.widget.ImageView;
import Android.widget.ViewSwitcher;
import Android.widget.Gallery.LayoutParams;
public class ImageShowActivity extends Activity implements
AdapterView.OnItemSelectedListener, ViewSwitcher.ViewFactory {
    private Integer[] mThumbIds= {
            R.drawable.image_thumb_0, R.drawable.image_thumb_1,
            R.drawable.image_thumb_2, R.drawable.image_thumb_3,
            R.drawable.image_thumb_4};
    private Integer[] mImageIds= {
            R.drawable.image_0, R.drawable.image_1, R.drawable.image_2,
            R.drawable.image_3, R.drawable.image_4};
    private ImageSwitcher mSwitcher;
@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.image_show);
        setTitle("ImageShowActivity");
        mSwitcher= (ImageSwitcher) findViewById(R.id.switcher);
        mSwitcher.setFactory(this);
        mSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
                Android.R.anim.fade_in));
        mSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
                Android.R.anim.fade_out));
        Gallery g= (Gallery) findViewById(R.id.gallery);
        g.setAdapter(new ImageAdapter(this));
        g.setOnItemSelectedListener(this);
}
public void onItemSelected(AdapterView parent, View v, int position, long id) {
        mSwitcher.setImageResource(mImageIds[position]);
}
public void onNothingSelected(AdapterView parent) {
}
public View makeView() {
        ImageView i= new ImageView(this);
        i.setBackgroundColor(0xFF000000);
        i.setScaleType(ImageView.ScaleType.FIT_CENTER);
        i.setLayoutParams(new ImageSwitcher.LayoutParams(LayoutParams.FILL_
        PARENT,
                LayoutParams.FILL_PARENT));
        return i;
}
public class ImageAdapter extends BaseAdapter {
```




```
private Context mContext;
        public ImageAdapter(Context c) {
            mContext= c;
        }
        public int getCount() {
            return mThumbIds.length;
        }
        public Object getItem(int position) {
            return position;
        }
        public long getItemId(int position) {
            return position;
        }
       public View getView(int position, View convertView, ViewGroup parent) {
            ImageView i= new ImageView(mContext);
            i.setImageResource(mThumbIds[position]);
            i.setAdjustViewBounds(true);
            i.setLayoutParams(new Gallery.LayoutParams(
                    LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
            i.setBackgroundResource(R.drawable.picture_frame);
            return i;
        }
    }
}
```

3.11　DigitalClock

Android.widget.DigitalClock 是数字时钟，继承自 Android.widget.TextView，在 Android.widget 包中。

用法如下：调用方法 getText()，可得到时间字符串。使用方法 addTextChanged-Listener(Android.text.TextWatcher)添加文字更改监听，则会每秒钟激发一次事件。

举例说明，界面如图 3.17 所示。

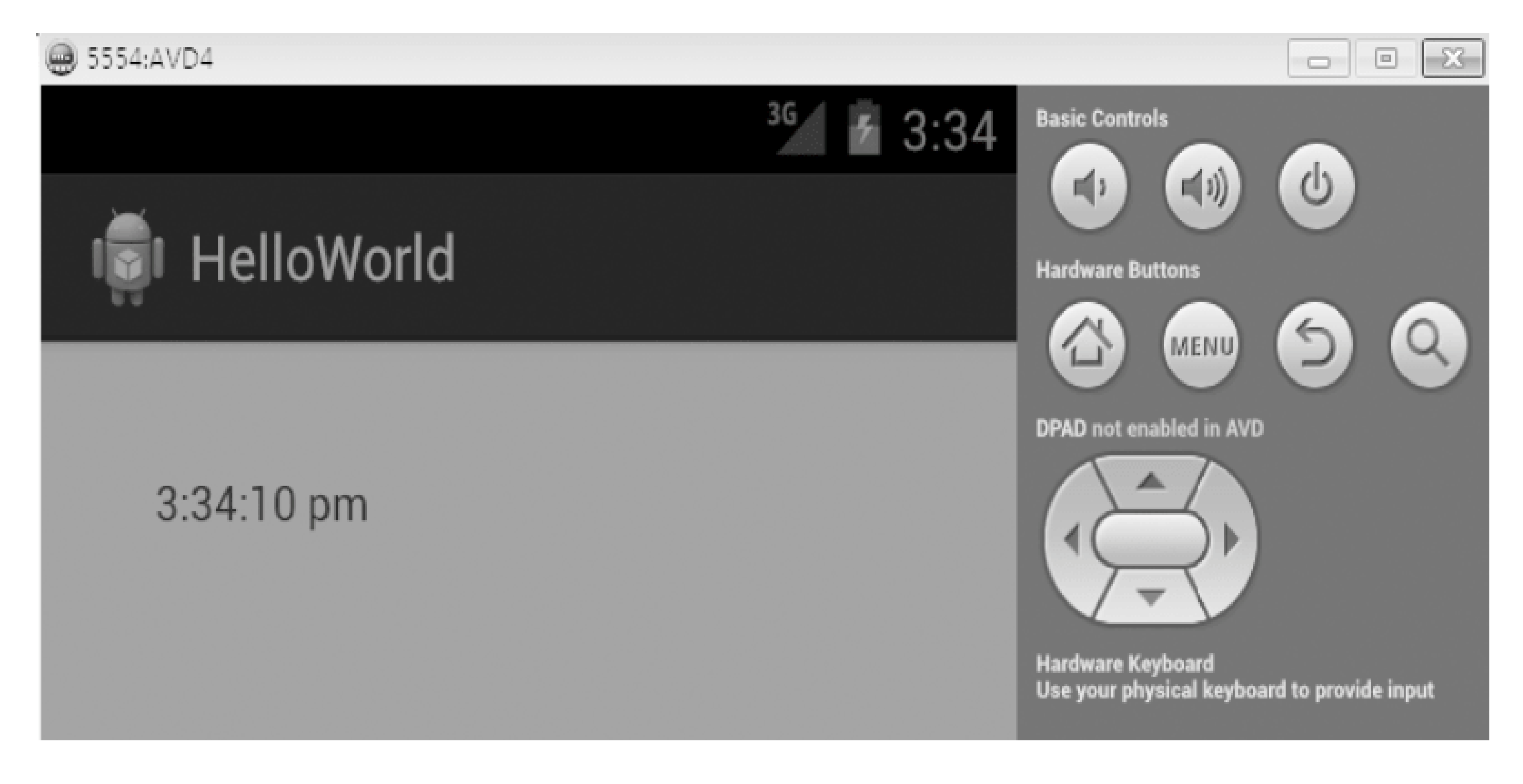


图 3.17　DigitalClock

该界面布局的 XML 代码如下：

```
<DigitalClock
        Android:id="@+id/digitalClock"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignParentLeft="true"
        Android:layout_alignParentTop="true"
        Android:layout_marginLeft="21dp"
        Android:layout_marginTop="20dp"
        Android:text="DigitalClock" />
```

该界面对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.widget.DigitalClock;
public class DigitalClockActivity extends Activity {
    DigitalClock clock;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_digitalclock);
        clock=(DigitalClock)super.findViewById(R.id.digitalClock);
        String strdate=clock.getText().toString();
    }
}
```

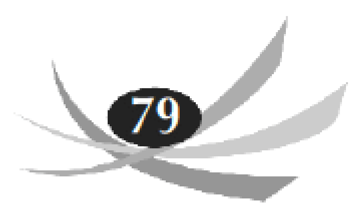
3.12 AnalogClock

Android.widget.AnalogClock 是模拟时钟，继承自 Android.view.View，在 Android.widget 包中。在界面中显示一个带时针和分针的模拟时钟。

举例说明，界面如图 3.18 所示。

该界面布局的 XML 代码如下：

```
<AnalogClock
        Android:id="@+id/analogClock"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignParentLeft="true"
        Android:layout_alignParentTop="true" />
    <DigitalClock
        Android:id="@+id/digitalClock"
        Android:layout_width="wrap_content"
```



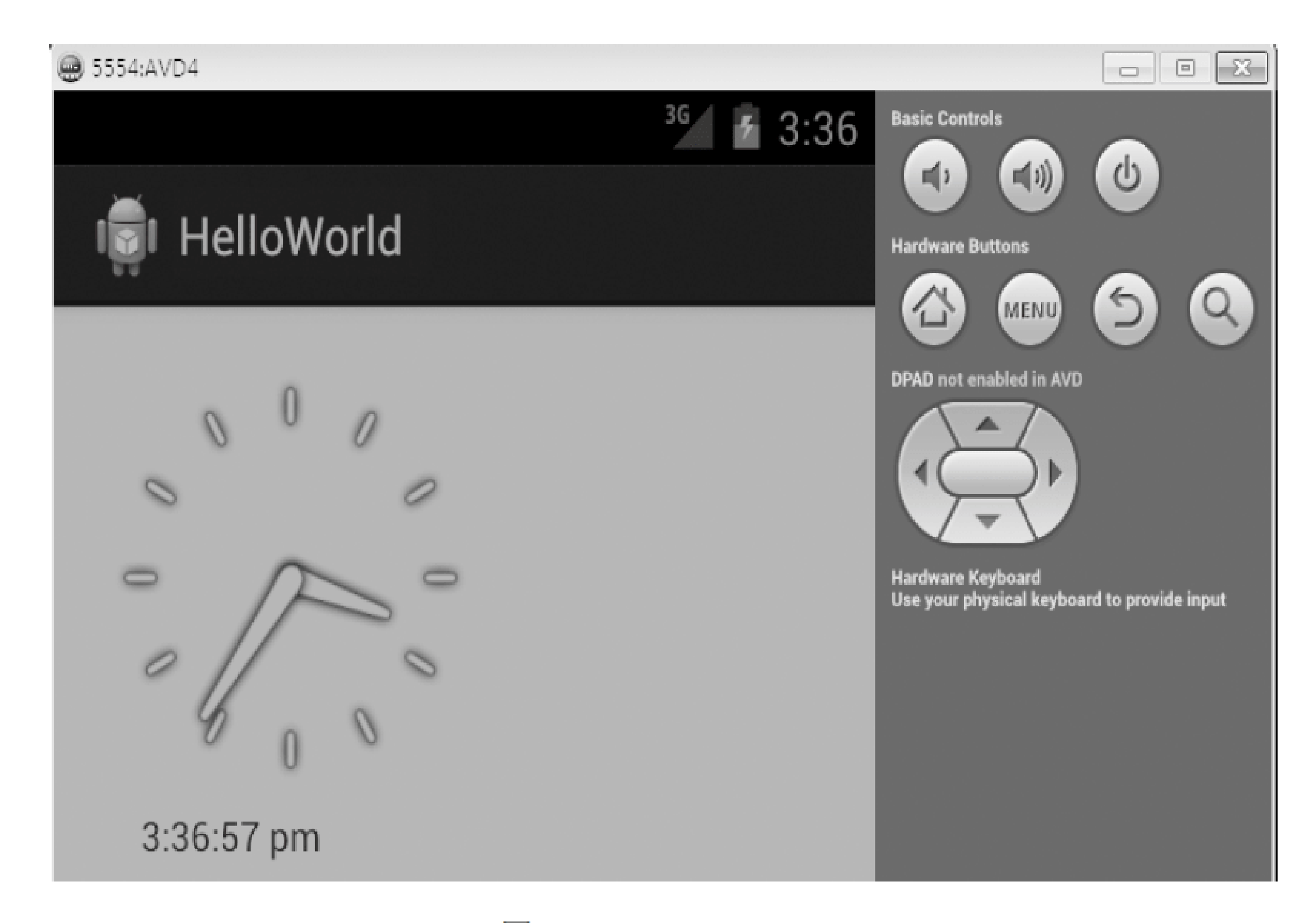


图 3.18　AnalogClock

```
        Android:layout_height="wrap_content"
        Android:layout_alignLeft="@+id/analogClock"
        Android:layout_below="@+id/analogClock"
        Android:layout_marginLeft="18dp"
        Android:text="DigitalClock" />
```

该界面对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.widget.AnalogClock;
import Android.widget.DigitalClock;
public class AnalogClockActivity extends Activity {
    AnalogClock clock;
    DigitalClock txtclock;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_analogclock);
        clock= (AnalogClock)super.findViewById(R.id.analogClock);
        txtclock= (DigitalClock)super.findViewById(R.id.digitalClock);
    }
}
```

3.13　TimePicker

Android.widget.TimePicker 是时间设置，继承自 Android.widget.FrameLayout (ViewGroup)，在 Android.widget 包中。

用法如下：

进行微调小时、分钟和 AM/PM(如果适用)。可以键盘输入，在 AM/PM 下单击按钮选择，也可以使用此视图的对话框 TimePickerDialog。

方法 getCurrentHour()得到当前小时(根据 AM/PM)。

方法 getCurrentMinute()得到当前分钟。

方法 setOnTimeChangedListener(TimePicker.OnTimeChangedListener)可对调节进行监听。

举例说明，界面如图 3.19 所示。



图 3.19 TimePicker

该界面布局的 XML 代码如下：

```
<TimePicker
  Android:id="@+id/time_picker"
  Android:layout_width="wrap_content"
  Android:layout_height="wrap_content"/>
```

该界面对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.widget.TimePicker;
public class TimePickerActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setTitle("TimePickerActivity");
        setContentView(R.layout.time_picker);
        TimePicker tp=    (TimePicker)this.findViewById(R.id.time_picker);
        tp.setIs24HourView(true);
    }
}
```

3.14　DatePicker

Android. widget. DatePicker 是时间设置，继承自 Android. widget. FrameLayout (ViewGroup)，在 Android. widget 包中。

用法如下：

进行微调年、月、日。可以键盘输入，也可以使用此视图的对话框 DatePickerDialog。

方法 getYear()得到年，getMonth()得到月，getDayOfMonth()得到当月日。

方法 init(int year，int monthOfYear，int dayOfMonth，DatePicker. OnDateChanged-Listener onDateChangedListener)初始化年月日和调节监听。

举例说明，界面如图 3.20 所示。



图 3.20　DatePicker

该界面布局的 XML 代码如下：

```
<DatePicker
        Android:id="@+id/date_picker"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content" />
```

该界面对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.widget.DatePicker;
public class DatePickerActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setTitle("CheckBoxActivity");
        setContentView(R.layout.date_picker);
        DatePicker dp=   (DatePicker)this.findViewById(R.id.date_picker);
        dp.init(2013, 1, 17, null);
    }
}
```

3.15 ProgressBar

Android. widget. ProgressBar 继承自 Android. view. View，在 Android. widget 包中，对应对话框 ProgressDialog。

ProgressBar 有两种展示方式：表盘形式(普通、小、大)和条形填充形式。在 layout 定义时，需要通过 style 属性类设置展示方式。progressBarStyleHorizontal 方式时，需要指定进度条最大值、当前值、次要的当前值。还可以在代码中手动设置。

常用属性设置如表 3-4 所示。

表 3-4　ProgressBar 属性

属　　性	说　　明
style	显示方式，取值： ? Android:attr/progressBarStyleLarge/progressBarStyle/progressBarStyleSmall/progressBarStyleHorizontal
Android:max	progressBarStyleHorizontal 方式时，进度条满时的值
Android:progress	progressBarStyleHorizontal 方式时，进度条主进度当前值
Android:secondaryProgress	progressBarStyleHorizontal 方式时，进度条次进度当前值

举例说明，界面如图 3.21 所示。



图 3.21　ProgressBar

该界面布局的 XML 代码如下：

```
<TextView
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:text="圆形进度条"/>
<ProgressBar
  Android:id="@+id/progress_bar"
  Android:layout_width="wrap_content"
  Android:layout_height="wrap_content"/>
```

```
<TextView
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:text="水平进度条"/>
<ProgressBar Android:id="@+id/progress_horizontal"
        style="?Android:attr/progressBarStyleHorizontal"
        Android:layout_width="200dip"
        Android:layout_height="wrap_content"
        Android:max="100"
        Android:progress="50"
        Android:secondaryProgress="75"/>
```

该界面对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.widget.ProgressBar;
public class ProgressBarActivity extends Activity {
    ProgressBar process;
    ProgressBar hprocess;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setTitle("ProgressBarActivity");
        setContentView(R.layout.progress_bar);
        process=(ProgressBar)super.findViewById(R.id.progress_bar);
        hprocess=
            (ProgressBar)super.findViewById(R.id.progress_horizontal);
    }
}
```

3.16 SeekBar

Android.widget.SeekBar 是拖动进度条，继承自 Android.widget.AbsSeekBar (Android.widget.ProgressBar)，在 Android.widget 包中。

常用属性设置如表 3-5 所示。

表 3-5 SeekBar 属性

属性	说明
Android:max	进度条满时的值
Android:progress	进度条主进度当前值
Android:thumb	拇指跟随图标
Android:thumbOffset	设置允许的轨道的范围扩展到拇指的拇指偏移量

方法 setOnSeekBarChangeListener(SeekBar.OnSeekBarChangeListener)可进行监听，监听内容包括开始拖曳、停止拖曳以及拖曳中进度条的值是否是用户改变等的参数改变。

举例说明，界面如图 3.22 所示。



图 3.22 SeekBar

该界面布局的 XML 代码如下：

```
<TextView
    Android:id= "@+ id/showseek"
        Android:layout_width= "wrap_content"
        Android:layout_height= "wrap_content"
        Android:text= "SeekBar" />
<SeekBar
        Android:id= "@+ id/seek"
        Android:layout_width= "fill_parent"
        Android:layout_height= "wrap_content"
        Android:max= "100"
        Android:thumb= "@ drawable/seeker"
        Android:progress= "50"/>
```

该界面对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.util.Log;
import Android.widget.SeekBar;
import Android.widget.TextView;

public class SeekBarActivity extends Activity {
    SeekBar bar;
    TextView showseek;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setTitle("SeekBarActivity");
        setContentView(R.layout.seek_bar);
```

```
            bar= (SeekBar)super.findViewById(R.id.seek);
            showseek= (TextView)super.findViewById(R.id.showseek);
            this.bar.setOnSeekBarChangeListener(onSeekBarChangeListener);
        }
        private SeekBar.OnSeekBarChangeListener onSeekBarChangeListener=
                    new SeekBar.OnSeekBarChangeListener() {
            public void onProgressChanged(SeekBar seekBar, int progress, boolean
                        fromUser) {
                showseek.setText(progress+ "");
                Log.v("", "SeekBar onProgressChanged, progress: "+ progress+
                    ", fromUser: "+ fromUser);
            }
            public void onStartTrackingTouch(SeekBar seekBar) {
                Log.v("", "SeekBar onStartTrackingTouch");
            }
            public void onStopTrackingTouch(SeekBar seekBar) {
                Log.v("", "SeekBar onStopTrackingTouch");
            }
        };
    }
```

3.17 RatingBar

Android.widget.RatingBar是星式进度条，继承自Android.widget.AbsSeekBar(Android.widget. ProgressBar)，在Android.widget包中。

常用属性设置如表3-6所示。

表3-6 RatingBar属性

属 性	说 明
Android:isIndicator	是否是评级栏，指示器作用。True为指示器，用户不可操作
Android:numStars	总星数
Android:rating	当前星数
Android:stepSize	每次可以等加的最小单位。浮点数

setOnRatingBarChangeListener(RatingBar.OnRatingBarChangeListener)添加一个监听器，可以监听每次改变。

举例说明，界面如图3.23所示。

该界面布局的XML代码如下：

```
<TextView
    Android:id= "@ + id/show_rating"
        Android:layout_width= "wrap_content"
        Android:layout_height= "wrap_content"
```

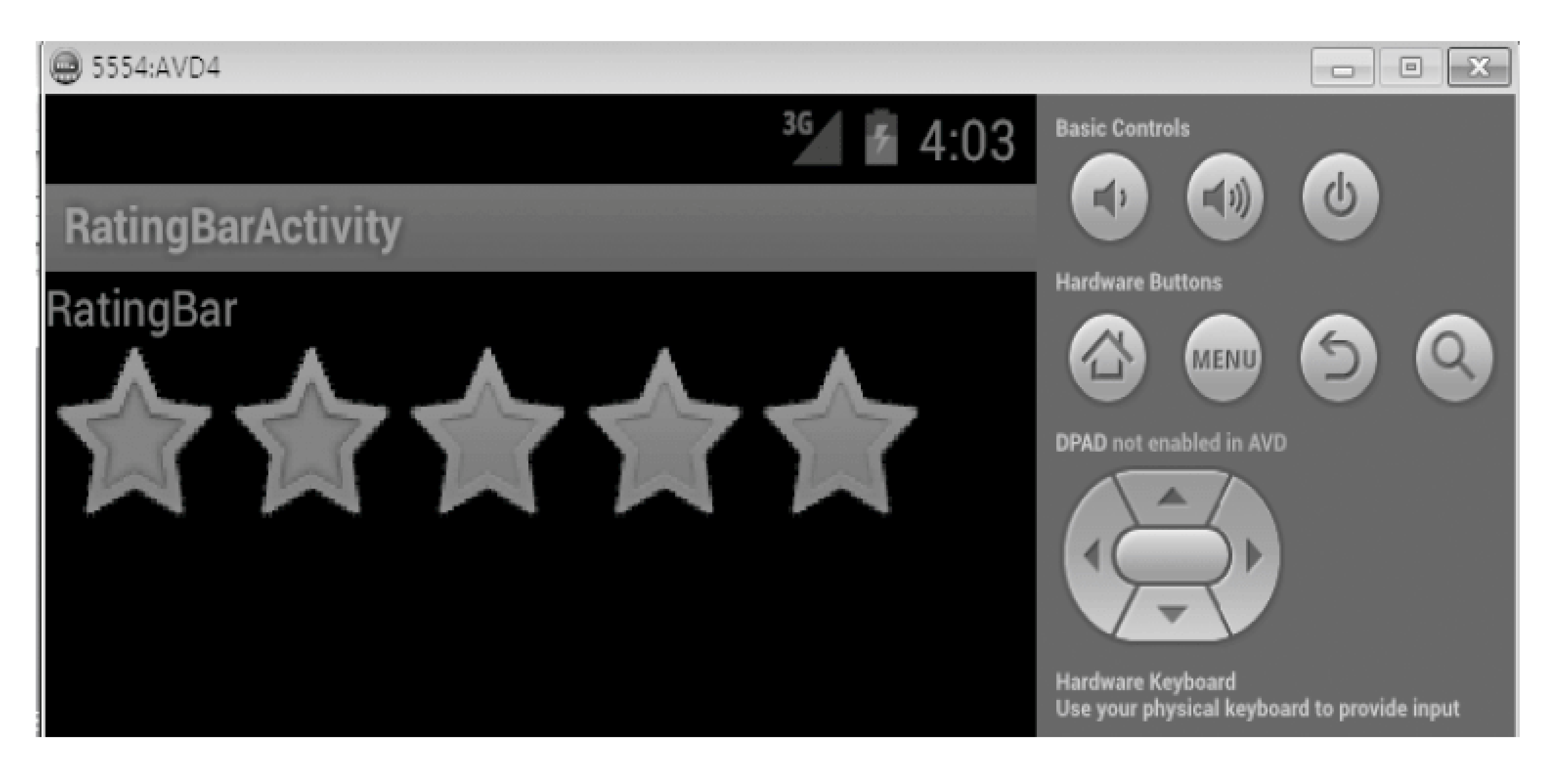
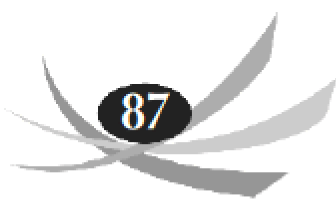


图 3.23 RatingBar

```
        Android:text= "RatingBar" />
  <RatingBar Android:id= "@+id/rating_bar"
  Android:layout_width= "wrap_content"
  Android:layout_height= "wrap_content"
  Android:rating= "1"/>
```

该界面对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.util.Log;
import Android.widget.RatingBar;
import Android.widget.TextView;
public class RatingBarActivity extends Activity {
    RatingBar bar;
    TextView showbar;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setTitle("RatingBarActivity");
        setContentView(R.layout.rating_bar);
        bar= (RatingBar)super.findViewById(R.id.rating_bar);
        showbar= (TextView)super.findViewById(R.id.show_rating);
        this.bar.setOnRatingBarChangeListener(onRatingBarChangeListener);
    }
    private RatingBar.OnRatingBarChangeListener onRatingBarChangeListener
                    = new RatingBar.OnRatingBarChangeListener() {
        public void onRatingChanged(RatingBar ratingBar, float rating,
                        boolean fromUser) {
            showbar.setText(rating+ " / "+ ratingBar.getNumStars());
            Log.v("", "RatingBar onRatingChanged, rating:  "+ rating+ ",
                        fromUser: "+ fromUser);
        }
    };
}
```



3.18　Spinner

Android.widget.Spinner 是下拉列表。下拉列表(Spinner)是一个每次只能选择所有项中一项的部件。

常用属性如表 3-7 所示。

表 3-7　Spinner 属性

属　性	功　能	说　明
Android:prompt	当 Spinner 对话框显示时显示的提示	是对其他资源的参照，形式为“@[+][package:]type:name”或“?[package:][type:]name”形式的主题属性

举例说明，界面如图 3.24 和图 3.25 所示。



图 3.24　Spinner(1)

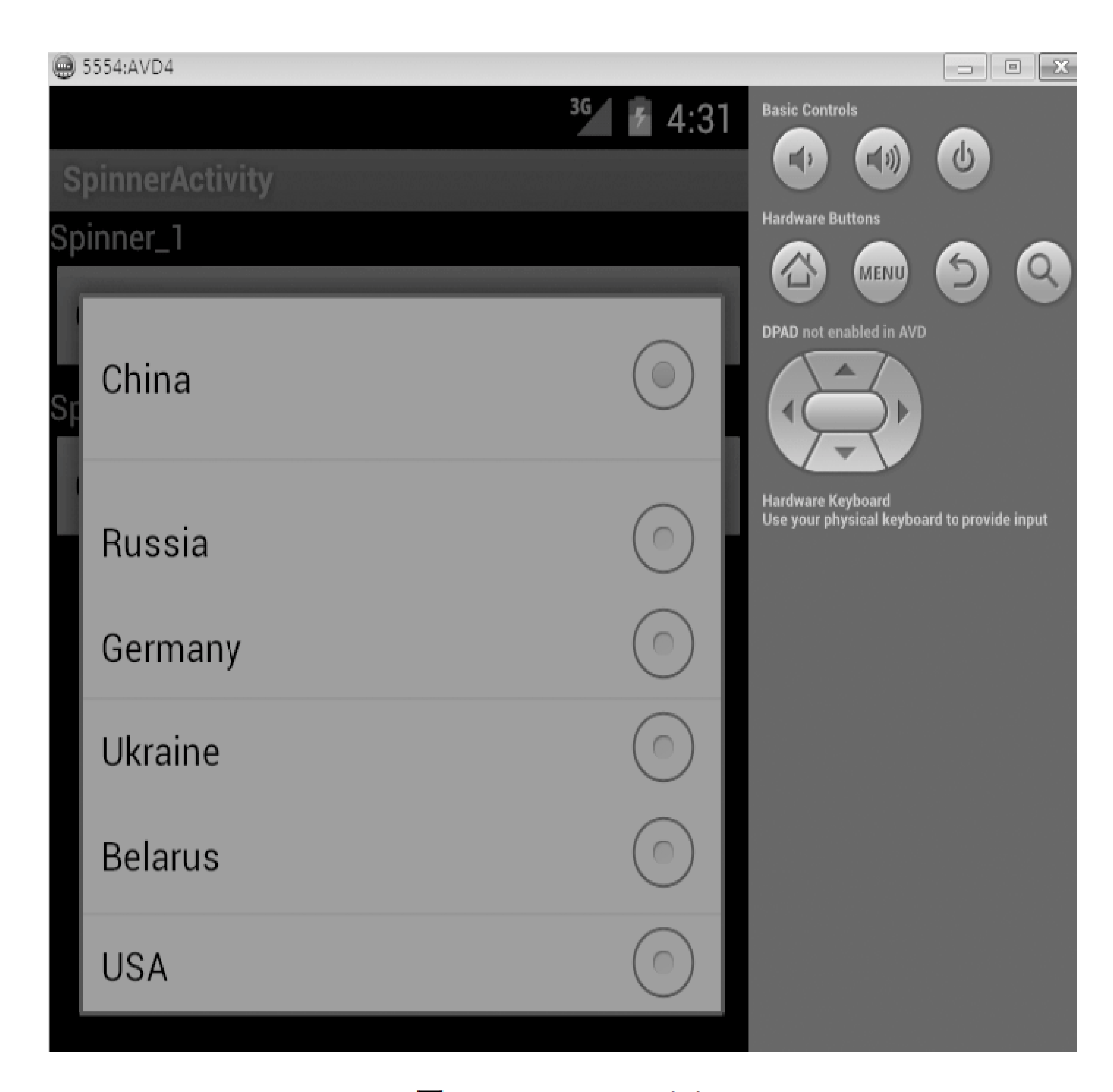


图 3.25　Spinner(2)

该界面布局的XML代码如下：

```
<TextView
    Android:layout_width="fill_parent"
    Android:layout_height="wrap_content"
    Android:text="Spinner_1"
    />
<Spinner  Android:id="@+id/spinner_1"
      Android:layout_width="fill_parent"
      Android:layout_height="wrap_content"
      Android:drawSelectorOnTop="false"/>
<TextView
    Android:layout_width="fill_parent"
    Android:layout_height="wrap_content"
    Android:text="Spinner_2 From arrays xml file"    />
<Spinner  Android:id="@+id/spinner_2"
      Android:layout_width="fill_parent"
      Android:layout_height="wrap_content"
      Android:drawSelectorOnTop="false"/>
```

该界面对应的Activity的Java代码如下：

```
import Java.util.ArrayList;
import Java.util.List;
import Android.app.Activity;
import Android.os.Bundle;
import Android.widget.ArrayAdapter;
import Android.widget.Spinner;
public class SpinnerActivity extends Activity {
    Spinner spinner_c;
    Spinner spinner_2;
    private ArrayAdapter<String> aspnCountries;
    private List<String> allcountries;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setTitle("SpinnerActivity");
        setContentView(R.layout.spinner);
        find_and_modify_view();
    }
    private static final String[] mCountries= { "China","Russia", "Germany",
            "Ukraine", "Belarus", "USA" };
    private void find_and_modify_view() {
        spinner_c= (Spinner) findViewById(R.id.spinner_1);
        allcountries= new ArrayList<String>();
```

```
            for(int i=0; i<mCountries.length; i++) {
                allcountries.add(mCountries[i]);
            }
            aspnCountries=new ArrayAdapter<String>(this,
                    Android.R.layout.simple_spinner_item, allcountries);
            aspnCountries.setDropDownViewResource
                        (Android.R.layout.simple_spinner_dropdown_item);
            spinner_c.setAdapter(aspnCountries);
            spinner_2=(Spinner) findViewById(R.id.spinner_2);
              ArrayAdapter<CharSequence> adapter=
                              ArrayAdapter.createFromResource(
                                         this, R.array.countries,
                                        Android.R.layout.simple_spinner_item);
            adapter.setDropDownViewResource
                              (Android.R.layout.simple_spinner_dropdown_item);
            spinner_2.setAdapter(adapter);
        }
}
```

3.19　实现注册界面

在本节中提供注册界面的代码，供大家参考。

界面布局的 XML 代码如下：

```
<TextView
        Android:id="@+id/reg_usernametitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_below="@+id/reg_imgtitle"
        Android:layout_marginLeft="5dp"
        Android:layout_marginTop="10dp"
        Android:text="用户名：" />
   <EditText
        Android:id="@+id/reg_username"
        Android:hint="请输入用户名"
        Android:textSize="12sp"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignBaseline="@+id/reg_usernametitle"
        Android:layout_alignBottom="@+id/reg_usernametitle"
        Android:layout_marginLeft="20dp"
        Android:layout_toRightOf="@+id/reg_usernametitle"
        Android:ems="10" />
```

```
<TextView
    Android:id="@+id/reg_passtitle"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_alignLeft="@+id/reg_usernametitle"
    Android:layout_below="@+id/reg_usernametitle"
    Android:layout_marginTop="10dp"
    Android:text="密码:"/>
<EditText
    Android:id="@+id/reg_pass"
    Android:hint="请输入密码"
    Android:inputType="textPassword"
    Android:textSize="12sp"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_alignBaseline="@+id/reg_passtitle"
    Android:layout_alignBottom="@+id/reg_passtitle"
    Android:layout_alignLeft="@+id/reg_username"
    Android:layout_toRightOf="@+id/reg_passtitle"
    Android:ems="10"/>
<TextView
    Android:id="@+id/reg_repasstitle"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_alignLeft="@+id/reg_usernametitle"
    Android:layout_below="@+id/reg_passtitle"
    Android:layout_marginTop="10dp"
    Android:text="再次输入:"/>
<EditText
    Android:id="@+id/reg_repass"
    Android:hint="请再输入一次密码"
    Android:textSize="12sp"
    Android:inputType="textPassword"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_alignBaseline="@+id/reg_repasstitle"
    Android:layout_alignBottom="@+id/reg_repasstitle"
    Android:layout_alignLeft="@+id/reg_username"
    Android:layout_toRightOf="@+id/reg_repasstitle"
    Android:ems="10"/>
<TextView
    Android:id="@+id/reg_emailtitle"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
```

```
        Android:layout_alignLeft="@+id/reg_usernametitle"
        Android:layout_below="@+id/reg_repasstitle"
        Android:layout_marginTop="10dp"
        Android:text="邮箱:"/>
    <EditText
        Android:id="@+id/reg_email"
        Android:hint="请输入邮箱"
        Android:textSize="12sp"
        Android:inputType="textEmailAddress"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignBaseline="@+id/reg_emailtitle"
        Android:layout_alignBottom="@+id/reg_emailtitle"
        Android:layout_alignLeft="@+id/reg_username"
        Android:layout_toRightOf="@+id/reg_emailtitle"
        Android:ems="10"/>
    <TextView
        Android:id="@+id/reg_phonetitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignLeft="@+id/reg_usernametitle"
        Android:layout_below="@+id/reg_emailtitle"
        Android:layout_marginTop="10dp"
        Android:text="手机:"/>
    <EditText
        Android:id="@+id/reg_phone"
        Android:hint="请输入手机号"
        Android:textSize="12sp"
        Android:inputType="phone"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignBaseline="@+id/reg_phonetitle"
        Android:layout_alignBottom="@+id/reg_phonetitle"
        Android:layout_alignLeft="@+id/reg_username"
        Android:layout_toRightOf="@+id/reg_phonetitle"
        Android:ems="10"/>
    <TextView
        Android:id="@+id/reg_niantitle"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_alignLeft="@+id/reg_usernametitle"
        Android:layout_below="@+id/reg_phonetitle"
        Android:layout_marginTop="10dp"
        Android:text="工作年数:"/>
```

```
<EditText
    Android:id="@+id/reg_nian"
    Android:hint="请输入工作年数"
    Android:textSize="12sp"
    Android:inputType="number"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_alignBaseline="@+id/reg_niantitle"
    Android:layout_alignBottom="@+id/reg_niantitle"
    Android:layout_alignLeft="@+id/reg_username"
    Android:layout_toRightOf="@+id/reg_niantitle"
    Android:ems="10" />
<TextView
    Android:id="@+id/reg_title"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_alignLeft="@+id/reg_username"
    Android:layout_alignParentTop="true"
    Android:layout_marginTop="20dp"
    Android:text="用户注册"
    Android:textSize="24sp" />
<Button
    Android:id="@+id/reg_ok"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_below="@+id/reg_nian"
    Android:layout_marginTop="60dp"
    Android:layout_toLeftOf="@+id/reg_nian"
    Android:text="注册" />
<Button
    Android:id="@+id/reg_cancel"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_alignBaseline="@+id/reg_ok"
    Android:layout_alignBottom="@+id/reg_ok"
    Android:layout_alignRight="@+id/reg_nian"
    Android:text="取消" />
<ImageView
    Android:id="@+id/reg_imgtitle"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_below="@+id/reg_title"
    Android:layout_margin="2dp"
    Android:layout_marginRight="14dp"
```

```
        Android:layout_toLeftOf="@+id/reg_cancel"
        Android:src="@drawable/ic_launcher" />
```

对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.Menu;
import Android.widget.Button;
import Android.widget.DatePicker;
import Android.widget.DatePicker.OnDateChangedListener;
import Android.widget.EditText;
import Android.widget.TimePicker;
import Android.widget.TimePicker.OnTimeChangedListener;
public class RegisterActivity   extends Activity {
    EditText username;
    EditText pass;
    EditText repass;
    EditText phone;
    EditText email;
    EditText worknian;
    Button bt_ok;
    Button bt_cancel;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        username= (EditText)super.findViewById(R.id.reg_username);
            pass= (EditText)super.findViewById(R.id.reg_pass);
            repass= (EditText)super.findViewById(R.id.reg_repass);
            phone= (EditText)super.findViewById(R.id.reg_phone);
            email= (EditText)super.findViewById(R.id.reg_email);
            worknian= (EditText)super.findViewById(R.id.reg_nian);
            bt_ok= (Button)super.findViewById(R.id.reg_ok);
            bt_cancel= (Button)super.findViewById(R.id.reg_cancel);
    }
}
```

第4章

Android 事件

通过第3章的学习,我们了解了 Android 界面的最常用的组件,本章将学习 Android 事件。通过本章的学习,大家可以完善第3章的界面的功能。

本章主要对事件的处理进行分析。Android 应用程序中事件的处理秉承了 JavaSE 图形用户界面的处理方式和风格。

Android 在事件处理过程中主要涉及3个概念。

(1) 事件:表示用户在图形界面的操作的描述,通常是封装成各种类,例如,键盘操作相关事件类为 KeyEvent,触摸屏相关事件类为 MotionEvent。

(2) 事件源:指事件发生的场所,通常指各个控件,例如 Button、EditText 等控件。

(3) 事件处理者:指接收事件对象并对其进行处理的对象,事件处理一般是一个实现某些特定接口类创建的对象。

4.1 事件的过程及原理

4.1.1 事件的过程

(1) 为事件源对象添加监听,这样当某个事件被触发时,系统才会知道通知谁来处理该事件,如图4.1(a)所示。

(2) 当事件发生时,系统会将事件封装成相应类型的事件对象,并发送给注册到事件源的事件监听器,如图4.1(b)所示。

(3) 当监听器对象接收到事件对象之后,系统会调用监听器中相应的事件处理方法来处理事件并给出响应,如图4.1(c)所示。

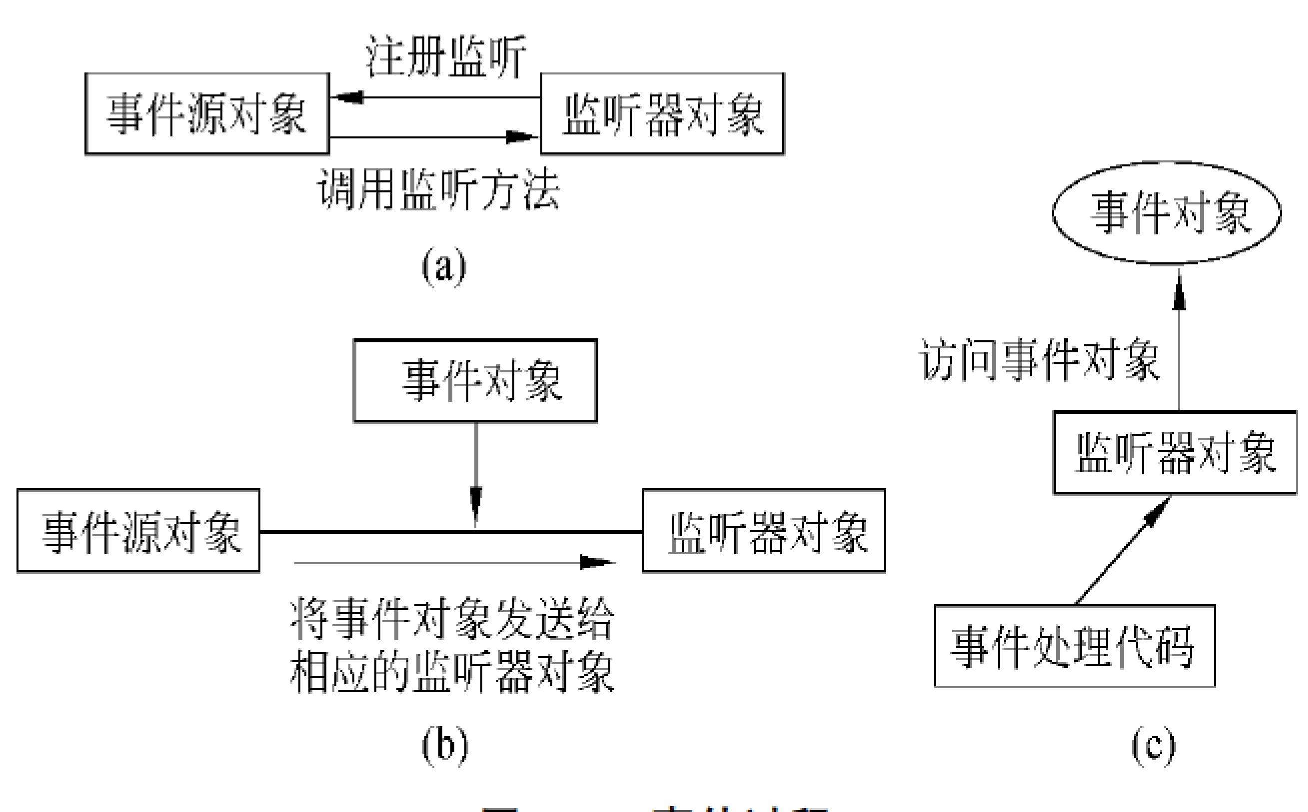


图4.1 事件过程

4.1.2　事件机制原理

通过 4.1.1 节事件过程原理的描述，我们大概能明白处理事件的一般步骤。下面就来分析系统是如何监听到一个输入设备事件的，又是利用何种方式将这种事件层层传递给应用层并作出响应的。这是个看似简单，而实际浩大的工程，我们且称之为事件系统或事件机制。

图 4.2 为 Android 事件大体传递过程。

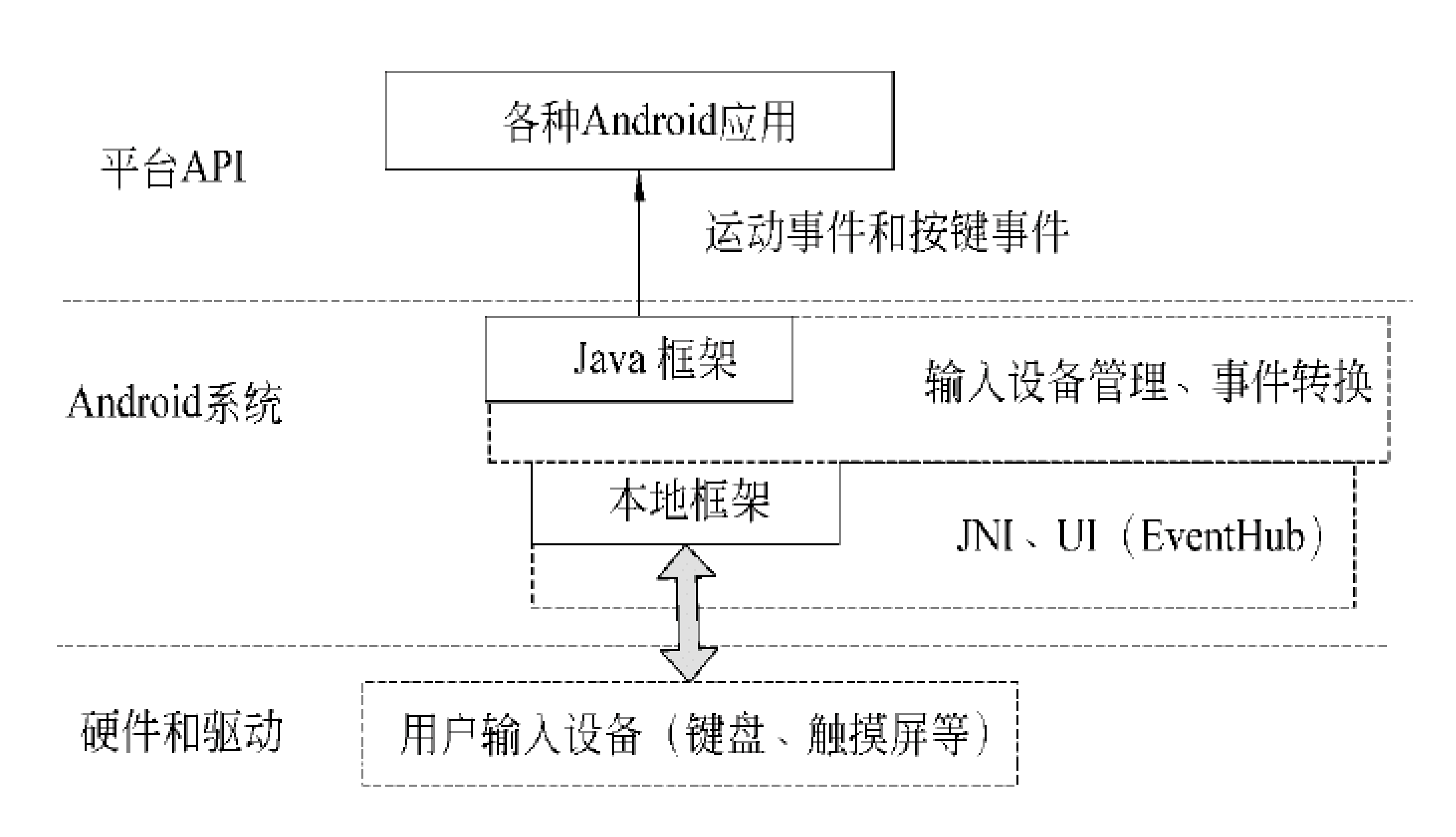


图 4.2　事件传递过程

图 4.3 为 Android 事件具体实现过程。

对图 4.3 自下而上层层剖析如下：

(1) 系统在启动过程中会加载驱动程序。

(2) 加载驱动程序时会进入 EventHub（文件位置：/frameworks/base/libs/ui/EventHub.cpp)，执行 openPlatformInput(void)这一函数（函数位置：文件的第 421～454 行），该函数的主要功能是扫描/dev/input 目录，获取输入设备（/dev/input 下通常是关于 Event 类型的驱动设备）。通过不断循环读取目录文件，最后通过 open_device()打开设备。

(3) EventHub 实现了对驱动程序的控制，并从中获得信息 KeyLayout（按键布局）和 KeyCharacterMap（按键字符映射）。定义按键布局和按键字符映射需要运行时配置文件的支持，它们的后缀名分别为.kl 和.kcm。

(4) 在 framework 层具有 KeyInputDevice 等类用于处理由 EventHub 传送上来的信息，通常信息由数据结构 RawInputEvent 和 KeyEvent 来表示。通常情况下，对于按键事件，直接使用 KeyEvent 来传送给应用程序层；对于触摸屏和轨迹球等事件，则由 RawInputEvent 经过转换后，形成 MotionEvent 传送给应用程序层。

(5) 在 Android 的应用程序层中，通过重新实现 onTouchEvent 和 onTrackballEvent 等函数来接收运动事件（MotionEvent），通过重新实现 onKeyDown 和 onKeyUp 等函数来接收按键事件（KeyEvent）。这些类包含在 Android.view 包中。

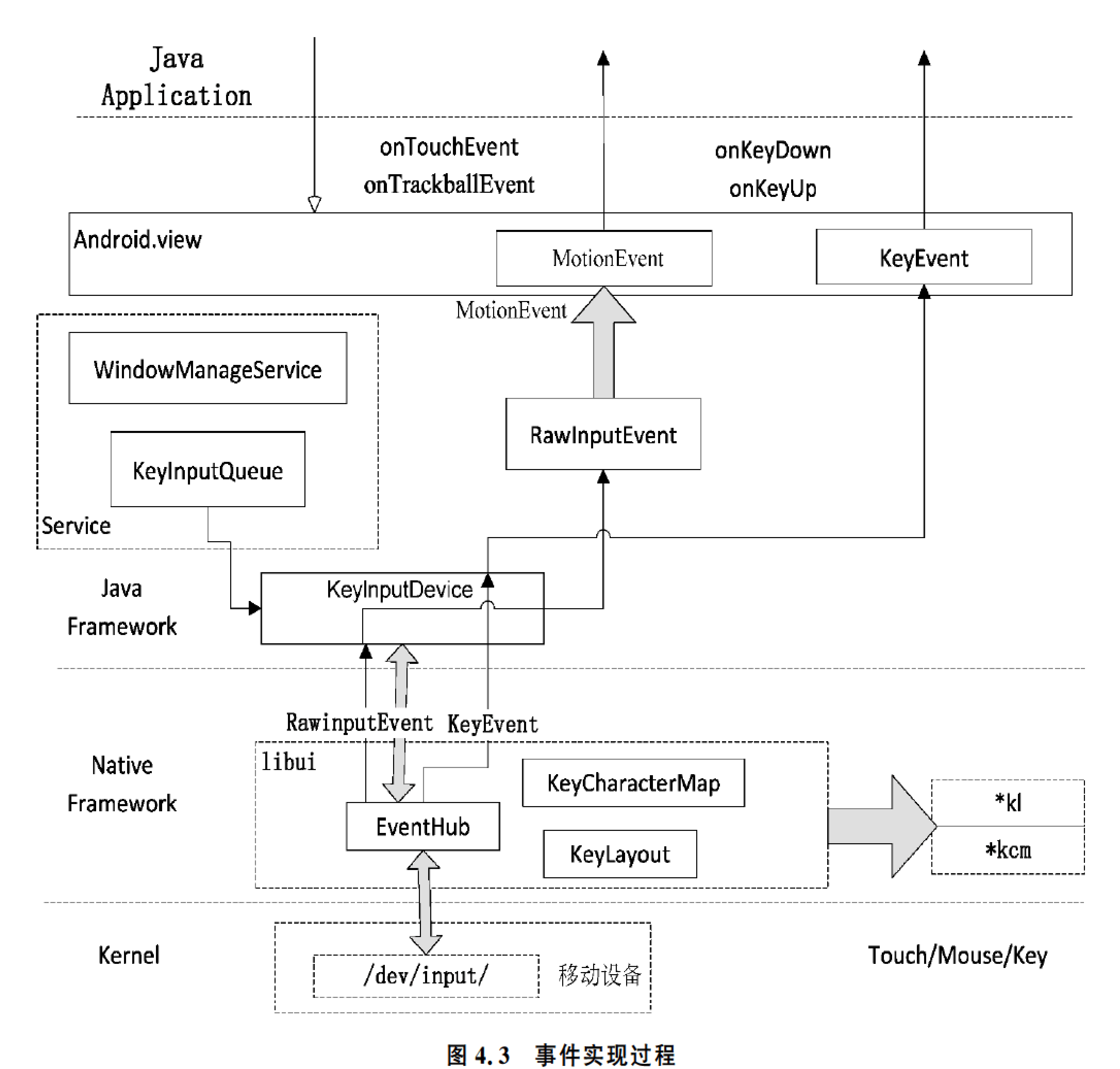


图 4.3 事件实现过程

4.2 事件处理模型

事件处理模型通常有 3 种方式：接口实现事件处理模型、内部类事件处理模型和匿名内部类事件处理模型。下面以响应 Button 事件源的单击事件以及弹出响应结果为例说明以上 3 种事件处理模型。

Button 按钮响应机制为：单击 Button 时，屏幕上的 TextView 中输出“你好!这是单击事件 !触发的是 Button1!”。

初始界面如图 4.4 所示。

单击 Button1 按钮之后，效果如图 4.5 所示。

单击 Button2 按钮之后，效果如图 4.6 所示。

3 种处理模型布局的 XML 文件相同，如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    Android:id="@+id/userlayout"
    Android:layout_width="fill_parent"
```



图 4.4　初始界面



图 4.5　Button1 单击



图 4.6　Button2 单击

```
    Android:layout_height="fill_parent"
    Android:orientation="vertical"
    xmlns:Android="http://schemas.Android.com/apk/res/Android">
<TextView
    Android:id="@+id/mytext"
```

```
        Android:layout_width="165dp"
        Android:layout_height="38dp"
        Android:text="TextView" />
    <Button
        Android:id="@+id/mybutton1"
        Android:layout_width="172dp"
        Android:layout_height="52dp"
        Android:text="Button1" />
    <Button
        Android:id="@+id/mybutton2"
        Android:layout_width="172dp"
        Android:layout_height="52dp"
        Android:text="Button2" />
    </LinearLayout>
```

4.2.1 接口实现事件处理模型

这种模型的实现方法是：在 Activity 继承事件的接口。Button 事件源的单击事件的接口是 OnClickListener，实现的接口方法是 onClick。

Activity 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.View;
import Android.view.View.OnClickListener;
import Android.widget.Button;
import Android.widget.TextView;
public class EventAActivity extends Activity implements OnClickListener{
    TextView mytext;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.event_1);
        setTitle("OnClickListener");
        mytext= (TextView) findViewById(R.id.mytext);
        //定义 Button 按钮
        Button mybutton1= (Button) findViewById(R.id.mybutton1);
        //注册监听
        mybutton1.setOnClickListener(this);
        //定义 Button 按钮
        Button mybutton2= (Button) findViewById(R.id.mybutton2);
        //注册监听
        mybutton2.setOnClickListener(this);
    }
```

```
    /**
     * 事件处理的实现
     * /
    @Override
    public void onClick(View v) {
    //循环当前触发 Click事件的事件源
        //v.getId()获取事件源 ID
        switch(v.getId()) {
    //判断是否是 Button1按钮触发的事件
        case R.id.mybutton1:
            mytext.setText("你好!这是单击事件 !触发的是 Button1!");
            break;
    //判断是否是 Button2按钮触发的事件
        case R.id.mybutton2:
            mytext.setText("你好!这是单击事件 !触发的是 Button2!");
            break;
        }
    }
}
```

请注意,如果有多个按钮,必须判断当前触发的是哪个按钮。例如本例中的处理方式。

4.2.2 内部类事件处理模型

我们先看实现代码。

Activity 代码如下:

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.View;
import Android.view.View.OnClickListener;
import Android.widget.Button;
import Android.widget.TextView;
public class EventBActivity extends Activity {
    TextView mytext= null;
    /**
     * 事件处理的实现
     * /
    class Clicklistener implements OnClickListener {
        @Override
        public void onClick(View v) {
            //循环当前触发 Click事件的事件源
            //v.getId()获取事件源 ID
            switch(v.getId()) {
```

```
                //判断是否是 Button1 按钮触发的事件
                case R.id.mybutton1:
                    mytext.setText("你好!这是单击事件!触发的是 Button1!");
                    break;
                //判断是否是 Button2 按钮触发的事件
                case R.id.mybutton2:
                    mytext.setText("你好!这是单击事件!触发的是 Button2!");
                    break;
                }
            }
        }
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.event_2);
            setTitle("OnClickListener");
            mytext= (TextView) findViewById(R.id.mytext);
            //定义 Button 按钮
            Button mybutton1= (Button) findViewById(R.id.mybutton1);
            //注册监听
            mybutton1.setOnClickListener(new Clicklistener());
            //定义 Button 按钮
            Button mybutton2= (Button) findViewById(R.id.mybutton2);
            //注册监听
            mybutton2.setOnClickListener(new Clicklistener());
        }
    }
```

这种实现模型和4.2.1节所讲的实现模型非常相似,只不过接口实现模型中是直接继承接口 OnClickListener,内部类实现模型是使用内部类继承 OnClickListener 接口。

备注:ClickListener 类也可以做成一个普通的类,不一定必须做成内部类,做成普通类完全与内部类效果一样。

4.2.3 匿名内部类事件处理模型

我们先看实现代码。

Activity 代码如下:

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.View;
import Android.view.View.OnClickListener;
import Android.widget.Button;
import Android.widget.TextView;
public class EventCActivity extends Activity {
```

```
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.event_3);
            final TextView mytext= (TextView) findViewById(R.id.mytext);
            setTitle("OnClickListener");
            //定义 Button1 按钮
            Button mybutton1= (Button) findViewById(R.id.mybutton1);
            //注册监听,绑定事件处理
            mybutton1.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    mytext.setText("你好!这是单击事件!触发的是 Button1!");
                }
            });
            //定义 Button1 按钮
            Button mybutton2= (Button) findViewById(R.id.mybutton2);
            //注册监听,绑定事件处理
            mybutton2.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    mytext.setText("你好!这是单击事件!触发的是 Button2!");
                }
            });
        }
    }
```

这种实现模型和前两种实现模型差别较大。区别在于对于每一个按钮都独立实现一个匿名内部类。

通过以上 3 种方式我们可以看出,Button 的 Click 事件处理的模型都是实现接口 OnClickListener 中的 onClick 方法,并且绑定于特定的事件源,从而达到事件的处理。其中,接口实现方法和内部类都是通过继承接口 OnClickListener 中的 onClick 方法;匿名内部类实现方法则是通过覆盖 onClick 方法。

4.3 Android 事件处理机制

Android 平台的事件处理机制有两种:一种是基于回调机制的;一种是基于监听接口的。

4.3.1 回调机制

Android 平台中,每个 View 都有自己的处理事件的回调方法,开发人员可以通过重写 View 中的这些回调方法来实现需要的响应事件。当某个事件没有被任何一个 View

处理时，便会调用 Activity 中相应的回调方法。Android 提供了以下回调方法供用户使用。

1. onKeyDown

功能：该方法是接口 KeyEvent.Callback 中的抽象方法，所有的 View 全部实现了该接口并重写了该方法，该方法用来捕捉手机键盘被按下的事件。

声明：public boolean onKeyDown (int keyCode, KeyEvent event)

参数说明如下。

(1) 参数 keyCode：该参数为被按下的键值即键盘码，手机键盘中每个按钮都会有其单独的键盘码，在应用程序中都是通过键盘码知道用户按下的是哪个键。

(2) 参数 event：该参数为按键事件的对象，其中包含了触发事件的详细信息，例如事件的状态、事件的类型、事件发生的时间等。当用户按下按键时，系统会自动将事件封装成 KeyEvent 对象供应用程序使用。

(3) 返回值：该方法的返回值为一个 boolean 类型的变量，当返回 true 时，表示已经完整地处理了这个事件，并不希望其他的回调方法再次进行处理；而当返回 false 时，表示并没有完全处理该事件，更希望其他回调方法继续对其进行处理，例如 Activity 中的回调方法。

2. onKeyUp

功能：该方法同样是接口 KeyEvent.Callback 中的一个抽象方法，并且所有的 View 同样全部实现了该接口并重写了该方法，该方法用来捕捉手机键盘按键抬起的事件。

声明：public boolean onKeyUp (int keyCode, KeyEvent event)

参数说明：同 onKeyDown。

3. onTouchEvent

功能：该方法在 View 类中定义，并且所有的 View 子类全部重写了该方法，应用程序可以通过该方法处理手机屏幕的触摸事件。

声明：public boolean onTouchEvent (MotionEvent event)

参数说明如下。

(1) 参数 event：该参数为手机屏幕触摸事件封装类的对象，其中封装了该事件的所有信息，例如触摸的位置、触摸的类型以及触摸的时间等。该对象会在用户触摸手机屏幕时被创建。

(2) 返回值：该方法的返回值机理与键盘响应事件的相同，同样是当已经完整地处理了该事件且不希望其他回调方法再次处理时返回 true，否则返回 false。

备注：该方法并不像之前介绍过的方法只处理一种事件，一般情况下以下 3 种情况的事件全部由 onTouchEvent 方法处理，只是 3 种情况中的动作值不同。

(1) 屏幕被按下：当屏幕被按下时，会自动调用该方法来处理事件，此时 MotionEvent.getAction()的值为 MotionEvent.ACTION_DOWN，如果在应用程序中需要处理屏幕被按下的事件，只需重新回调该方法，然后在方法中进行动作的判断即可。

(2) 屏幕被抬起：当触控笔离开屏幕时触发的事件，该事件同样需要 onTouchEvent 方法来捕捉，然后在方法中进行动作判断。当 MotionEvent.getAction()的值为

MotionEvent. ACTION_UP 时，表示是屏幕被抬起的事件。

(3) 在屏幕中拖动：该方法还负责处理触控笔在屏幕上滑动的事件，同样是调用 MotionEvent. getAction()方法来判断动作值是否为 MotionEvent. ACTION_MOVE 再进行处理。

注意：ACTION_DOWN 事件作为起始事件，它的重要性是超过 ACTION_MOVE 和 ACTION_UP 的，如果发生这两个事件，那么一定曾经发生了 ACTION_DOWN 事件。

4. onTrackBallEven

功能：该方法是手机中轨迹球的处理方法。所有的 View 同样全部实现了该方法。

声明：public boolean onTrackballEvent (MotionEvent event)

参数说明如下。

(1) 参数 event：该参数为手机轨迹球事件封装类的对象，其中封装了触发事件的详细信息，同样包括事件的类型、触发时间等，一般情况下，该对象会在用户操控轨迹球时被创建。

(2) 返回值：该方法的返回值与前面介绍的各个回调方法的返回值机制完全相同。备注：轨迹球与手机键盘的区别如下所示：

① 某些型号的手机设计出的轨迹球会比只有手机键盘时更美观，可增添用户对手机的整体印象。

② 轨迹球使用更为简单，例如在某些游戏中使用轨迹球控制会更为合理。

③ 使用轨迹球会比键盘更为细化，即滚动轨迹球时，后台的表示状态的数值会变化得更细微、更精准。

提示：在模拟器运行状态下，可以通过 F6 键打开模拟器的轨迹球，然后便可以通过鼠标的移动来模拟轨迹球事件。

5. onFocusChanged

功能：前面介绍的各个方法都可以在 View 及 Activity 中重写，这里介绍的 onFocusChanged 方法却只能在 View 中重写。该方法是焦点改变的回调方法，当某个控件重写了该方法后，当焦点发生变化时，会自动调用该方法来处理焦点改变的事件。

声明：protected void onFocusChanged (boolean gainFocus, int direction, Rect previously FocusedRect)

参数说明如下。

(1) 参数 gainFocus：该参数表示触发该事件的 View 是否获得了焦点，当该控件获得焦点时，gainFocus 等于 true，否则等于 false。

(2) 参数 direction：该参数表示焦点移动的方向，用数值表示，有兴趣的读者可以重写 View 中的该方法打印该参数进行观察。

(3) 参数 previouslyFocusedRect：该参数表示在触发事件的 View 的坐标系中，前一个获得焦点的矩形区域，即表示焦点是从哪里来的。如果不可用则为 null。

提示：焦点描述了按键事件(或者是屏幕事件等)的承受者，每次按键事件都发生在拥有焦点的 View 上。在应用程序中，可以对焦点进行控制，例如从一个 View 移动到另一个 View。下面列出一些与焦点有关的常用方法。

(1) setFocusable 方法：设置 View 是否可以拥有焦点。

(2) isFocusable 方法：监测此 View 是否可以拥有焦点。

(3) setNextFocusDownId 方法：设置 View 的焦点向下移动后获得焦点 View 的 ID。

(4) hasFocus 方法：返回 View 的父控件是否获得了焦点。

(5) requestFocus 方法：尝试让此 View 获得焦点。

(6) isFocusableTouchMode 方法：设置 View 是否可以在触摸模式下获得焦点，在默认情况下是不可能获取焦点的。

4.3.2 监听机制

下面就几个典型的监听事件接口进行说明。

1. OnClickListener 接口

功能：该接口处理的是单击事件。在触摸模式下，是在某个 View 上按下并抬起的组合动作；而在键盘模式下，是某个 View 获得焦点后单击确定键或者按下轨迹球事件。

对应的回调方法：public void onClick(View v)

说明：需要实现 onClick 方法，参数 v 是事件发生的事件源。

2. OnLongClickListener 接口

功能：该接口与 OnClickListener 接口原理基本相同，只是该接口是 View 长按事件的捕捉接口，即当长时间按下某个 View 时触发的事件。

对应的回调方法：public boolean onLongClick(View v)

说明：需要实现 onLongClick 方法。

参数 v：该参数为事件源控件，当长时间按下此控件时才会触发该方法。

返回值：该方法的返回值是一个 boolean 类型的变量，当返回 true 时，表示已经完整地处理了这个事件，并不希望其他的回调方法再次进行处理；当返回 false 时，表示并没有完全处理该事件，更希望其他方法继续对其进行处理。

3. OnFocusChangeListener 接口

功能：该接口用来处理控件焦点发生改变的事件。如果注册了该接口，当某个控件失去焦点或者获得焦点时都会触发该接口中的回调方法。

对应的回调方法：public void onFocusChange(View v, Boolean hasFocus)

说明：需要实现 onFocusChange 方法。

参数 v：该参数是触发该事件的事件源。

参数 hasFocus：该参数表示 v 的新状态，即 v 是否获得焦点。

4. OnKeyListener 接口

功能：该接口是对手机键盘进行监听的接口，通过对某个 View 注册该监听，当 View 获得焦点并有键盘事件时，便会触发该接口中的回调方法。

对应的回调方法：public boolean onKey(View v, int keyCode, KeyEvent event)

说明：需要实现 onKey 方法。

参数 v：该参数为事件的事件源控件。

参数 keyCode：该参数为手机键盘的键盘码。

参数 event：该参数是键盘事件封装类的对象，其中包含了事件的详细信息，例如发生的事件、事件的类型等。

5. OnTouchListener 接口

功能：该接口是用来处理手机屏幕事件的监听接口，当在 View 的范围内触摸按下、抬起或滑动等动作时都会触发该事件。

对应的回调方法：public boolean onTouch(View v, MotionEvent event)

说明：需要实现 onTouch 方法。

参数 v：该参数同样为事件源对象。

参数 event：该参数为事件封装类的对象，其中封装了触发事件的详细信息，同样包括事件的类型、触发时间等信息。

6. OnCreateContextMenuListener 接口

功能：该接口是用来处理上下文菜单显示事件的监听接口。该方法是定义和注册上下文菜单的另一种方式。

对应的回调方法：public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo info)

说明：需要实现 onCreateContextMenu 方法。

参数 menu：该参数为事件的上下文菜单。

参数 v：该参数为事件源 View，当该 View 获得焦点时才可能接收该方法的事件响应。

参数 info：info 对象中封装了有关上下文菜单额外的信息，这些信息取决于事件源 View。

该方法会在某个 View 中显示上下文菜单时被调用，开发人员可以通过实现该方法来处理上下文菜单显示时的一些操作，其使用方法与前面介绍的各个监听接口没有任何区别。

4.4 常见事件

下面将对 Android 中常见的事件进行分析，常见的事件包括触摸屏事件、键盘事件和菜单事件。

对于触摸屏事件(MotionEvent)，有按下、弹起、移动、双击、长按、滑动、滚动。按下、弹起、移动是简单的触摸屏事件，而双击、长按、滑动、滚动需要根据运动的轨迹来做识别。

对于键盘事件(KeyEvent)，有按下、弹起、长按等。

对于菜单事件，则是菜单的关闭、显示等。

备注：Android 手机的坐标系是以左上定点为原点坐标(0,0)，向右为 X 轴正方向，向下为 Y 轴正方向。

4.4.1 触摸屏事件

在 Android 中任何一个控件和 Activity 都是间接或者直接继承于 Android.view.

View。一个 View 对象可以处理测距、布局、绘制、焦点变换、滚动条，以及触屏区域自己表现的按键和手势。触摸屏事件主要是通过运行事件(MotionEvent)接收触摸信息并进行处理的。如果触摸事件源是 Activity，则需要重写方法 onTouchEvent(MotionEvent)。

以下代码处理的事件来源主要包括按钮和触摸，Activity 程序如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.MotionEvent;
import Android.view.View;
import Android.view.View.OnClickListener;
import Android.widget.Button;
import Android.widget.TextView;
public class EventDActivity extends Activity {
    TextView mytext1= null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.event_4);
        setTitle("onTouchEvent ");
        mytext1= (TextView) findViewById(R.id.mytext1);
    //定义 Button 按钮
        Button mybutton1= (Button) findViewById(R.id.mybutton1);
        Button mybutton2= (Button) findViewById(R.id.mybutton2);
    //Button 按钮事件处理
        mybutton2.setOnClickListener(new clicklistener());
        mybutton1.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                mytext1.setText("触发事件的是 mybutton1");
            }
        });
    }
    class clicklistener implements OnClickListener {
        @Override
        public void onClick(View v) {
            if(v.getId()==R.id.mybutton2)
                mytext1.setText("触发事件的是 mybutton2");
        }
    }
    //触摸按钮以外区域处理,重写 onTouchEvent 方法
    @Override
    public boolean onTouchEvent(MotionEvent event)
    {
        TextView mytext2= (TextView)findViewById(R.id.mytext2);
```

```
            int Action= event.getAction();
            float x= event.getX();
            float y= event.getY();
            mytext1.setText("现在处理的是滑动,Action is:"+ Action);
            mytext2.setText("移动坐标 Postion is:"+ "("+ x+ ","+ y+ ")");
            return true;
        }
}
```

界面布局的 XML 文件如下:

```
<?xml version= "1.0" encoding= "utf-8"?>
<LinearLayout
    Android:id= "@+ id/mylayout"
    Android:layout_width= "fill_parent"
    Android:layout_height= "fill_parent"
    Android:orientation= "vertical"
    xmlns:Android= "http://schemas.Android.com/apk/res/Android">
<TextView
    Android:id= "@+ id/mytext1"
    Android:layout_width= "165dp"
    Android:layout_height= "38dp"
    Android:text= "mytext1" />
<Button
    Android:id= "@+ id/mybutton1"
    Android:layout_width= "150dp"
    Android:layout_height= "52dp"
    Android:text= "Button1" />
<Button
    Android:id= "@+ id/mybutton2"
    Android:layout_width= "150dp"
    Android:layout_height= "wrap_content"
    Android:text= "Button2" />
<TextView
    Android:id= "@+ id/mytext2"
    Android:layout_width= "fill_parent"
    Android:layout_height= "wrap_content"
    Android:text= "mytext2"/>
</LinearLayout>
```

处理过程界面如下。

开始界面如图 4.7 所示。

按钮单击事件界面如图 4.8 和图 4.9 所示。

在虚拟机按住鼠标左键滑动,响应如图 4.10 所示。



图 4.7　初始界面



图 4.8　Button1 单击

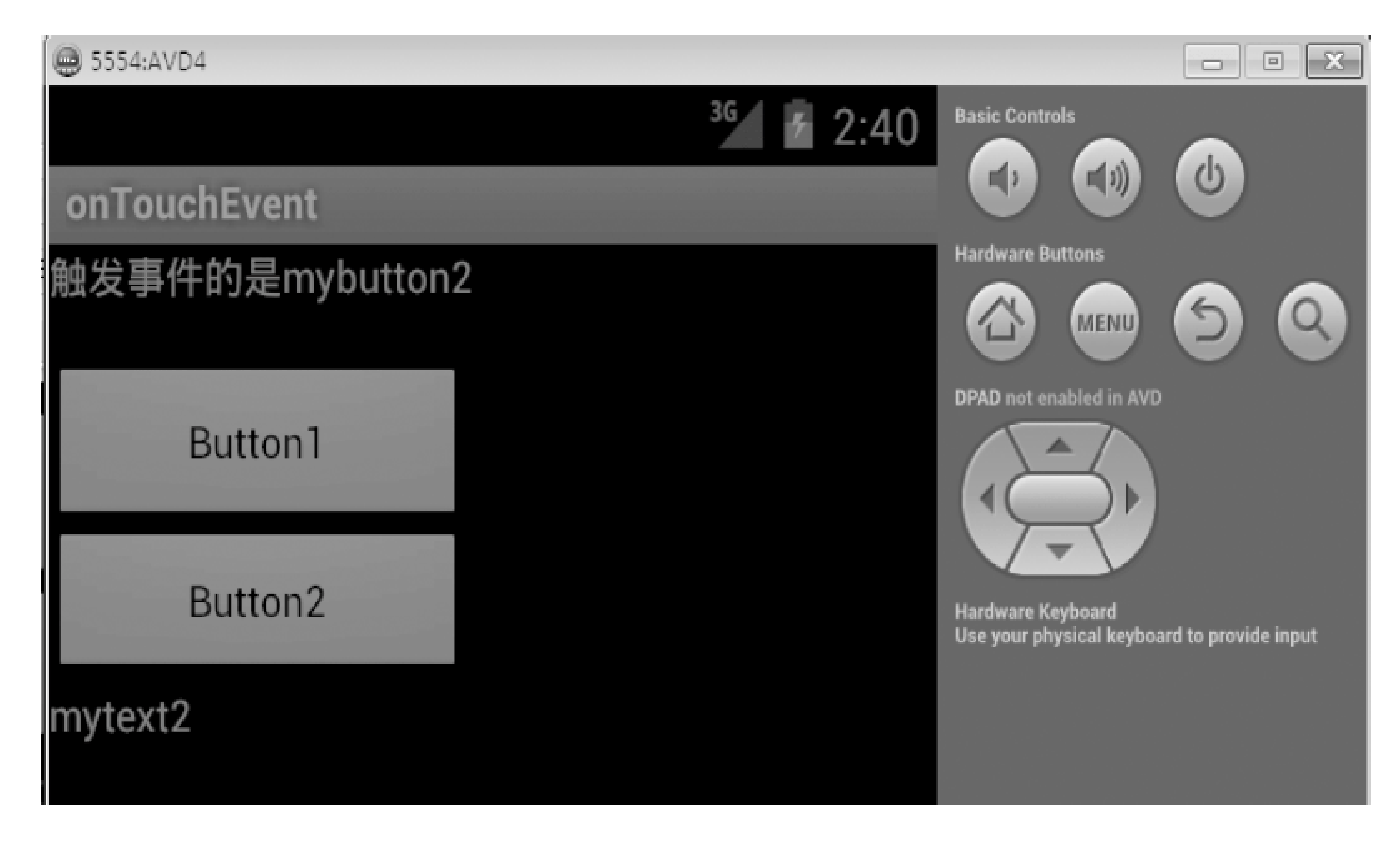


图 4.9　Button2 单击



图 4.10 触摸处理

4.4.2 手势识别

一个操作性优良的用户界面能够吸引用户的眼球。现在我们经常看到一些好的界面都带有滑动、滚动等效果。但是触摸屏是不产生滚动、滑动的消息的，需要根据其运动的轨迹用算法去判断实现。在 Android 系统中，使用 Android.view.GestureDetector 来实现手势的识别，我们只需要实现 GestureDetector 的 OnGestureListener 接口来侦听 GestureDetector 识别后的事件。我们需要在 GestureDetector 的 onTouchEvent 方法中进行轨迹识别。

以下代码展示手势识别，Activity 程序如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.GestureDetector;
import Android.view.GestureDetector.OnGestureListener;
import Android.view.MotionEvent;
import Android.widget.TextView;
public class EventEActivity extends Activity {
    TextView    m_eventshow;
    TextView    m_eventshow2;
    int oldevent=-1;
    private GestureDetector gestureDetector=new GestureDetector(new
                OnGestureListener()        {
        //鼠标按下的时候,会产生 onDown。由一个 ACTION_DOWN 产生
        public boolean onDown(MotionEvent event) {
            DisplayEventType("手势   mouse down"+" X="+event.getX()+",
                        Y="+event.getY());
            return false;
        }
        //用户按下触摸屏、快速移动后松开,这个时候,你的手指运动是有加速度的
```

```
        //由一个 MotionEvent ACTION_DOWN、多个 ACTION_MOVE、一个 ACTION_UP 触发
        //e1: 第一个 ACTION_DOWN MotionEvent
        //e2: 最后一个 ACTION_MOVE MotionEvent
        //velocityX: X 轴上的移动速度,像素/秒
        //velocityY: Y 轴上的移动速度,像素/秒
        public boolean onFling(MotionEvent e1, MotionEvent e2, float
                velocityX,float velocityY) {
            DisplayEventType("手势   onFling");
            return false;
        }
        //用户长按触摸屏,由多个 MotionEvent ACTION_DOWN 触发
        public void onLongPress(MotionEvent event) {
            DisplayEventType("手势   on long pressed");
        }
        //滚动事件,当在触摸屏上迅速地移动,会产生 onScroll。由 ACTION_MOVE 产生
        //e1: 第一个 ACTION_DOWN MotionEvent
        //e2: 最后一个 ACTION_MOVE MotionEvent
        //distanceX: 距离上次产生 onScroll 事件后,X 轴移动的距离
        //distanceY: 距离上次产生 onScroll 事件后,Y 轴移动的距离
        public boolean onScroll(MotionEvent e1, MotionEvent e2, float
                distanceX,float distanceY) {
            DisplayEventType("手势   onScroll"+ " X= "+ distanceX+ ",
                        Y= "+ distanceY);
            return false;
        }
        //单击了触摸屏,但是没有移动和弹起的动作。onShowPress 和 onDown 的区别在于
        //onDown 是一旦触摸屏按下,就马上产生 onDown 事件,但是 onShowPress 是
        //onDown 事件产生后,一段时间内,如果没有移动鼠标和弹起事件,就认为是
        //onShowPress 事件
        public void onShowPress(MotionEvent event) {
            DisplayEventType("手势   pressed");
        }
        //轻击触摸屏后,弹起。如果这个过程中产生了 onLongPress、onScroll 和
        //onFling 事件,就不会产生 onSingleTapUp 事件
        public boolean onSingleTapUp(MotionEvent event) {
            DisplayEventType("手势 Tap up!");
            return false;
        }
        public void DisplayEventType(String text){
            m_eventshow.setText(text);
        }
    });
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
            super.onCreate(savedInstanceState);
            setContentView(R.layout.event_5);
            setTitle("GestureDetector ");
            m_eventshow= (TextView)this.findViewById(R.id.mytext1);
            m_eventshow2= (TextView)this.findViewById(R.id.mytext2);
        }
        //上节中的重写 onTouchEvent
        @Override
        public boolean onTouchEvent(MotionEvent event)
        {
            m_eventshow2.setText("现在处理的 Action= :"+ event.getAction()+
                                 "X= "+ event.getX()+ ",Y= "+ event.getY());
            if(gestureDetector.onTouchEvent(event))
                return true;
            else
                return false;
        }
    }
```

对应的 Layout 的 XML 文件如下：

```
<?xml version= "1.0" encoding= "utf-8"?>
<LinearLayout xmlns:Android= "http://schemas.Android.com/apk/res/Android"
    Android:id= "@+ id/mylayout"
    Android:layout_width= "fill_parent"
    Android:layout_height= "fill_parent"
    Android:orientation= "vertical" >
    <TextView
        Android:id= "@+ id/mytext1"
        Android:layout_width= "277dp"
        Android:layout_height= "38dp"
        Android:text= "mytext1" />
    <TextView
        Android:id= "@+ id/mytext2"
        Android:layout_width= "277dp"
        Android:layout_height= "38dp"
        Android:text= "mytext2" />
</LinearLayout>
```

处理过程界面如下。

开始界面如图 4.11 所示，单击响应如图 4.12 所示，长按响应如图 4.13 所示，按住滑动响应如图 4.14 所示，按住并滑动放开后响应如图 4.15 所示。

还有其他一些响应提示，就不再一一列举，大家可以实现这段代码后自己运行测试一下。

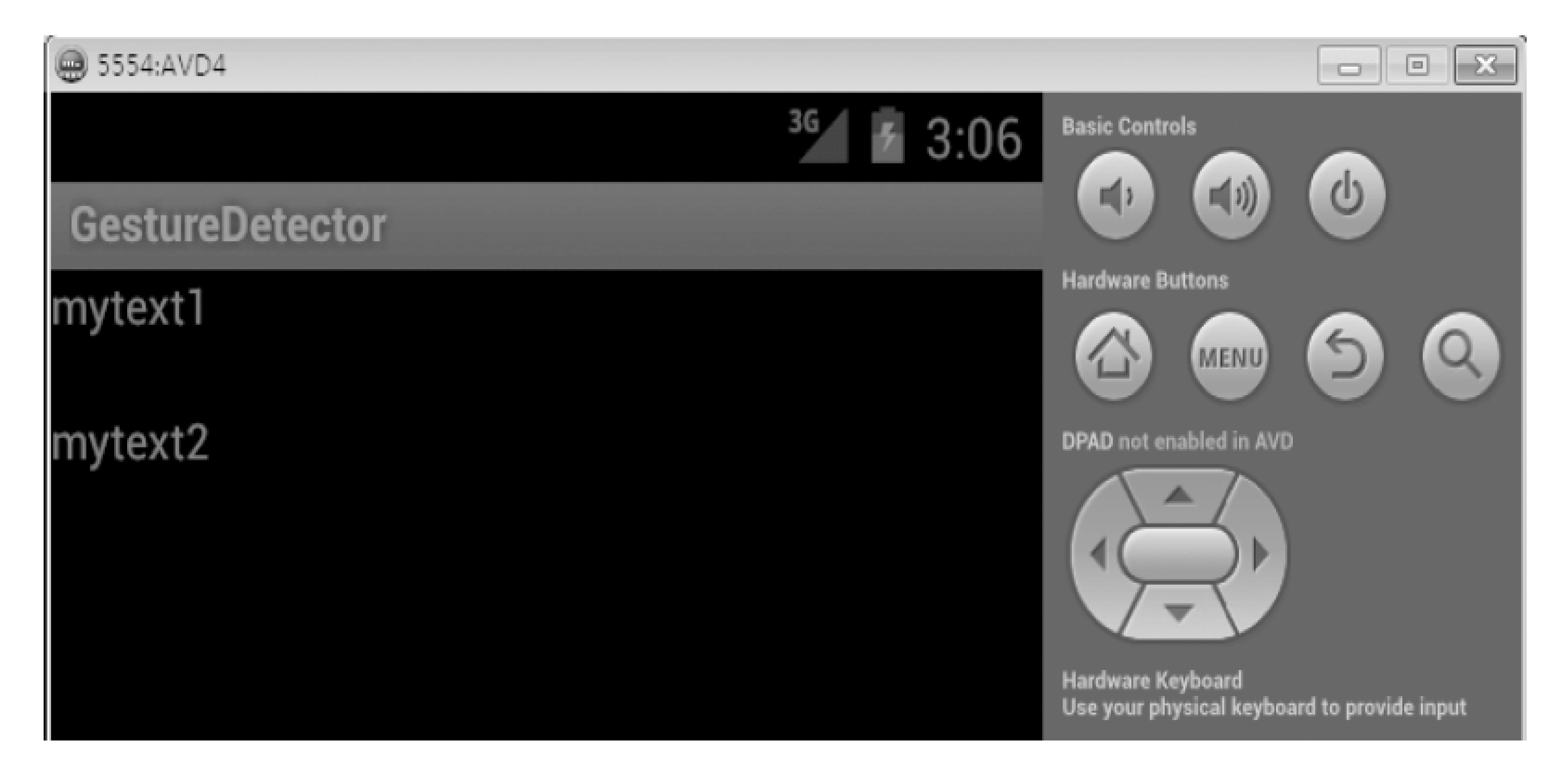


图 4.11　初始界面

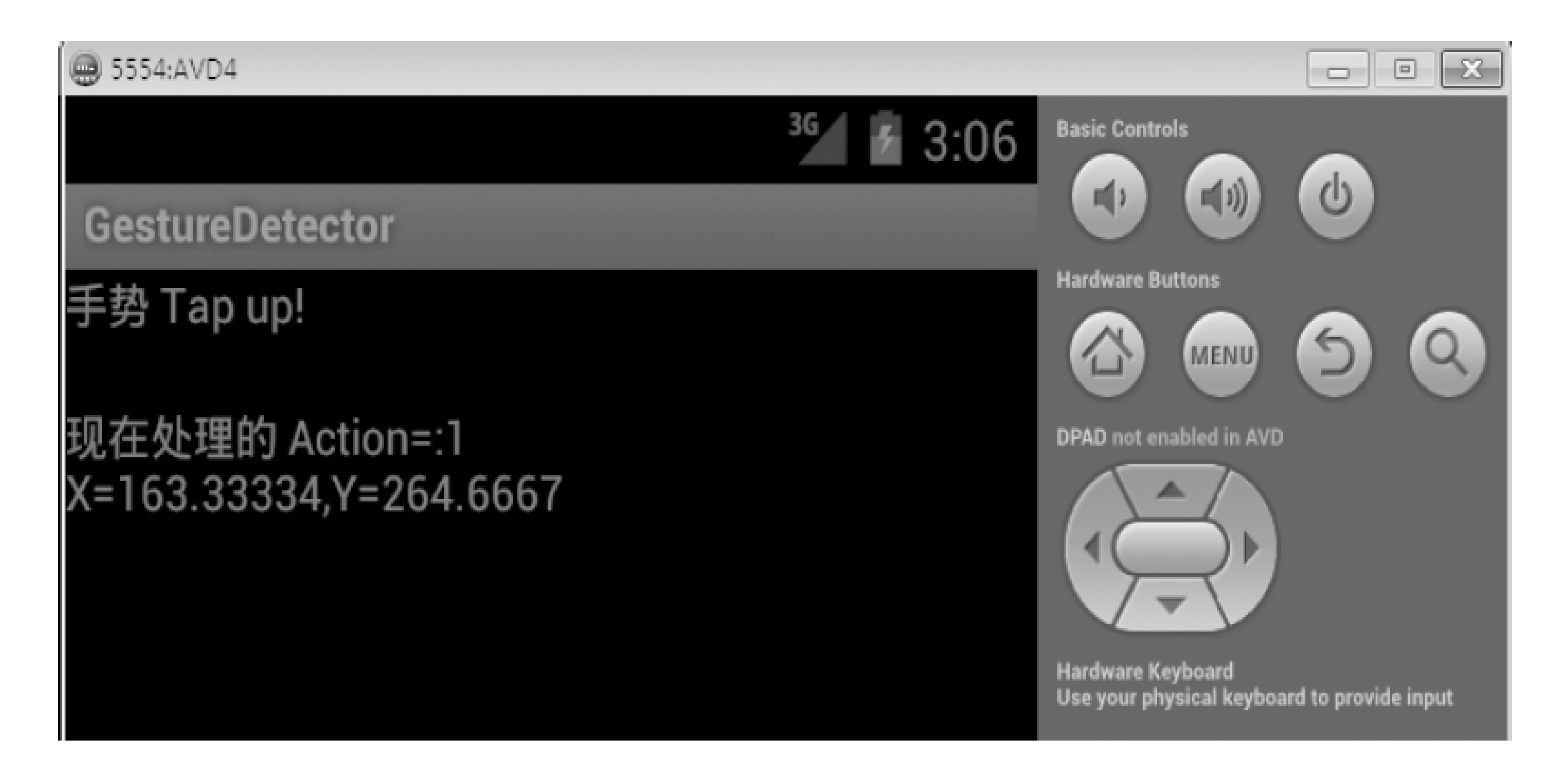


图 4.12　单击响应



图 4.13　长按响应



图 4.14　按住滑动响应



图 4.15　按住并滑动放开后响应

4.4.3　键盘事件

键盘事件是 Android 手机按钮的事件。

以下代码展示键盘事件,Activity 程序如下:

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.KeyEvent;
import Android.view.View;
import Android.view.View.OnKeyListener;
import Android.widget.EditText;
import Android.widget.TextView;

public class EventFActivity extends Activity {
    TextView m_eventShow;
    EditText m_editText;
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.event_6);
        setTitle("onKeyDown ");
        m_eventShow= (TextView)this.findViewById(R.id.mytext1);
        m_editText= (EditText)this.findViewById(R.id.editText1);
        m_editText.setOnKeyListener(new OnKeyListener(){
            @Override
            public boolean onKey(View v, int keyCode, KeyEvent event) {
                switch(event.getAction()) {
                  case KeyEvent.ACTION_UP:              //键盘松开
                      DisplayEventType(m_editText.getText().toString());
                      break;
                  case KeyEvent.ACTION_DOWN:            //键盘按下
                      DisplayEventType("ACTION_DOWN");
                      break;
                  }
                return false;
            }

        });
    }
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event)
     {
        switch(keyCode)
         {
            case KeyEvent.KEYCODE_HOME:
                DisplayEventType("Home down");
                      break;
            case KeyEvent.KEYCODE_BACK:
                DisplayEventType("Back down");
                break;
            case KeyEvent.KEYCODE_DPAD_LEFT:
                DisplayEventType("Left down");
                break;
        }
        return super.onKeyDown(keyCode, event);
    }
    @Override
    public boolean onKeyUp(int keyCode, KeyEvent event)
    {
        switch(keyCode)
        {
```

```
                case KeyEvent.KEYCODE_HOME:
                    DisplayEventType("Home up");
                    break;
                case KeyEvent.KEYCODE_BACK:
                        DisplayEventType("Back up");
                    break;
                case KeyEvent.KEYCODE_DPAD_LEFT:
                    DisplayEventType("Left up");
                    break;
            }
            return super.onKeyUp(keyCode, event);
        }
        public void DisplayEventType(String text){
                m_eventShow.setText(text);
        }
}
```

界面布局的 XML 文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    Android:id="@+id/mylayout"
    Android:layout_width="fill_parent"
    Android:layout_height="fill_parent"
    Android:orientation="vertical"
    xmlns:Android="http://schemas.Android.com/apk/res/Android">
<TextView
    Android:id="@+id/mytext1"
    Android:layout_width="165dp"
    Android:layout_height="38dp"
    Android:text="mytext1" />
<EditText
    Android:id="@+id/editText1"
    Android:layout_width="fill_parent"
    Android:layout_height="wrap_content"
    Android:ems="10" >
    <requestFocus />
</EditText>
</LinearLayout>
```

处理过程界面如下。

开始界面如图 4.16 所示，按返回按钮响应如图 4.17 所示，输入文字放开按键如图 4.18 所示，输入文字按下按键如图 4.19 所示。

在学习对话框后，使用对话框去展示对键盘事件的响应会更好展示。



图 4.16　起始界面



图 4.17　按返回按钮响应

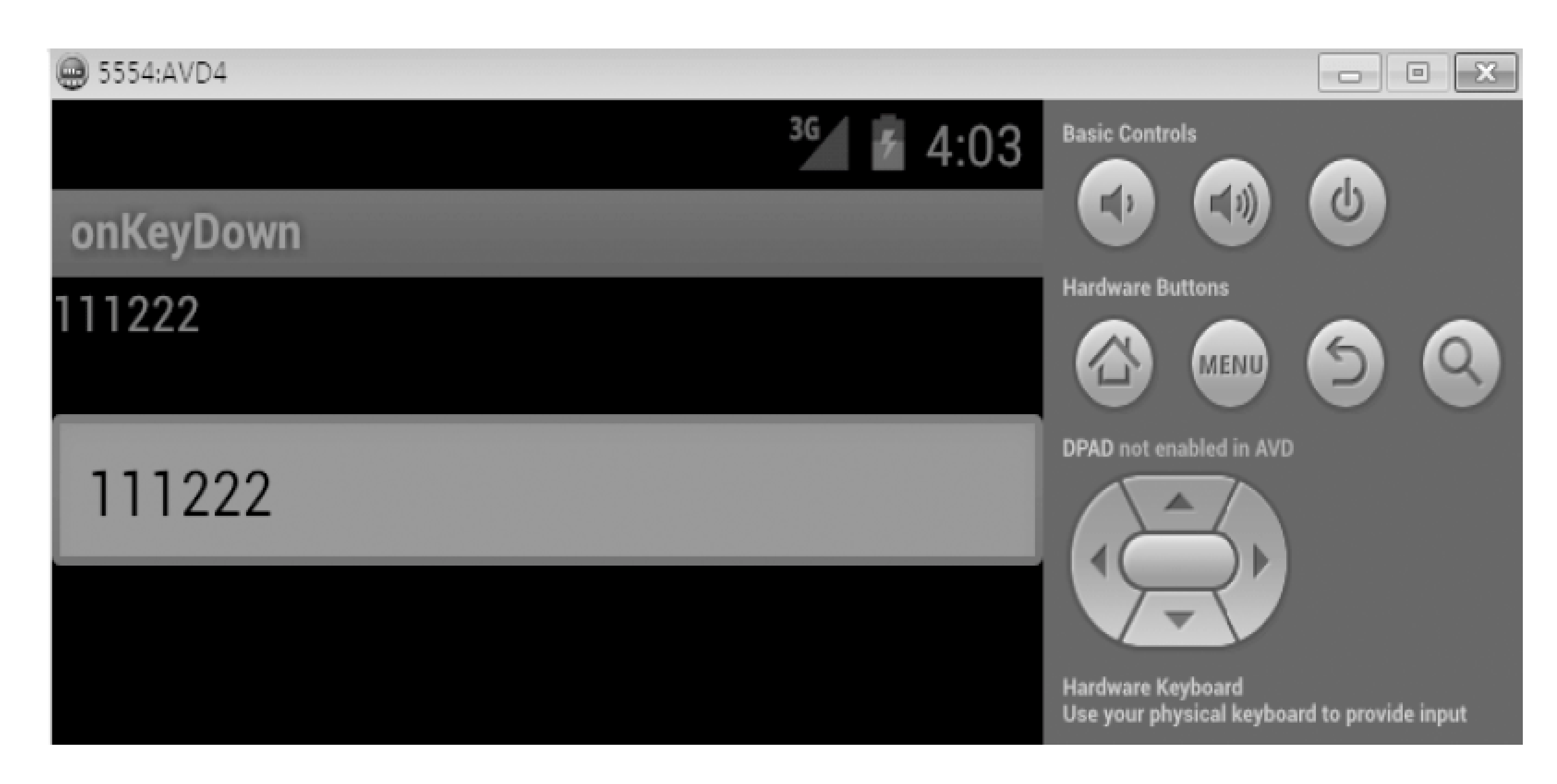


图 4.18　输入文字放开按键

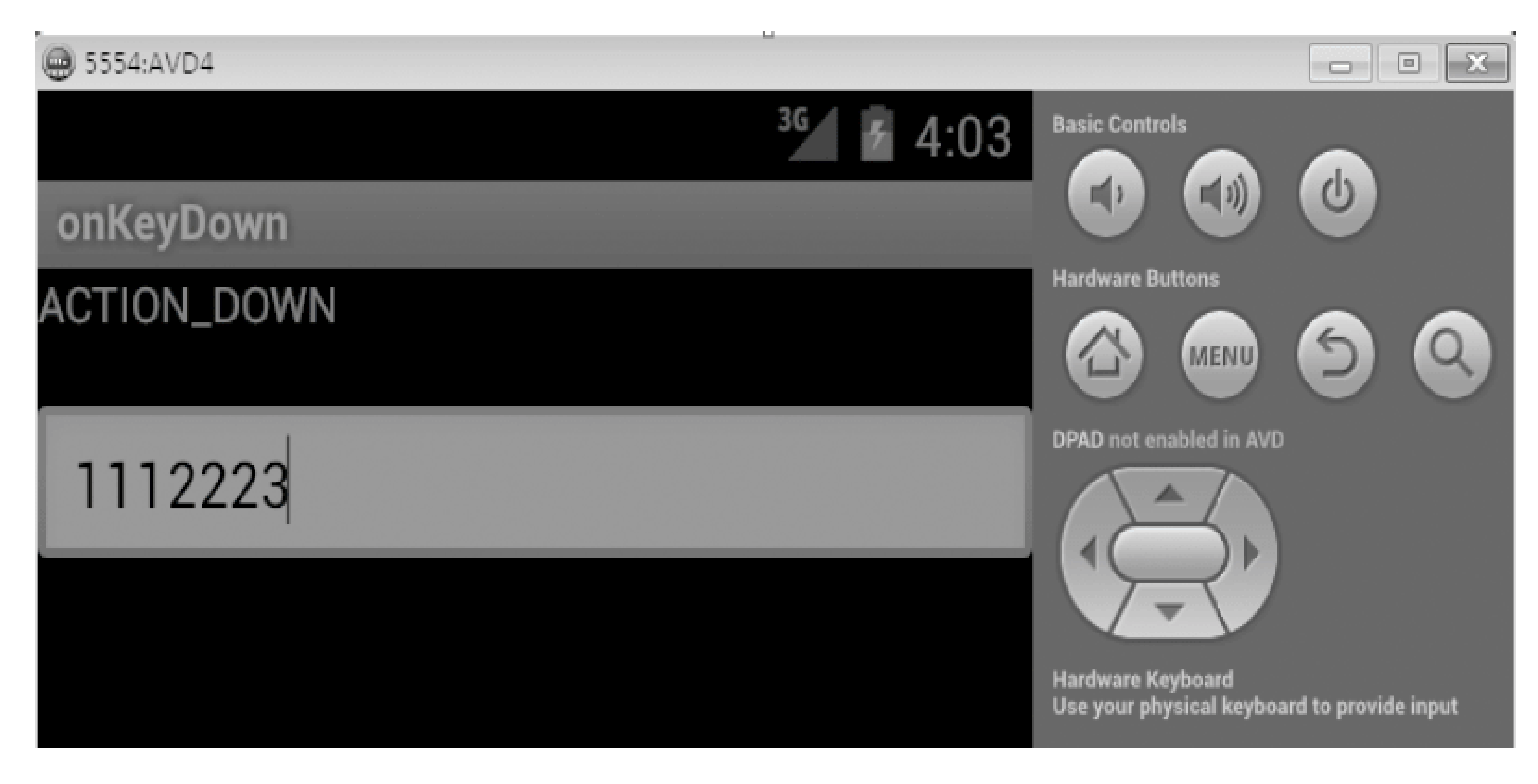


图 4.19 输入文字按下按键

4.4.4 模拟鼠标与按键事件

在程序开发中，经常会遇到需要由程序本身而非人为操作来发送鼠标或按键事件的情况。在 Android 中 Instrumentation 发送键盘鼠标事件。

Instrumentation 提供了丰富的以 send 开头的函数接口来实现模拟键盘鼠标，如下所述：

```
sendCharacterSync(int keyCode)          //用于发送指定 KeyCode 的按键
sendKeyDownUpSync(int key)              //用于发送指定 Key 的按键
sendPointerSync(MotionEvent event)      //用于模拟 Touch
sendStringSync(String text)             //用于发送字符串
```

以下代码展示模拟事件，Activity 程序如下：

```
import Android.app.Activity;
import Android.app.Instrumentation;
import Android.os.Bundle;
import Android.util.Log;
import Android.view.KeyEvent;
import Android.widget.TextView;
public class EventGActivity extends Activity {
    TextView m_eventshow;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.event_7);
        setTitle("onKeyDown ");
        m_eventshow= (TextView)this.findViewById(R.id.mytext1);
        moniKey(KeyEvent.KEYCODE_0);
        moniKey(KeyEvent.KEYCODE_N);
    }
    public static void moniKey(final int KeyCode) {
```

```
            new Thread() {
                    public void run() {
                        try {
                            Instrumentation inst= new Instrumentation();
                            inst.sendKeyDownUpSync(KeyCode);
                        } catch(Exception e) {
                            Log.e("Exception when sendKeyDownUpSync",
                                        e.toString());
                        }
                    }
                }.start();
            }
        @Override
        public boolean onKeyDown(int keyCode, KeyEvent event)
         {
            DisplayEventType(""+ keyCode);
            return super.onKeyDown(keyCode, event);
        }
        @Override
        public boolean onKeyUp(int keyCode, KeyEvent event)
        {
            DisplayEventType(""+ keyCode);
                return super.onKeyUp(keyCode, event);
        }
        public void DisplayEventType(String text){
                m_eventshow.setText(text);
        }
}
```

对应的 Layout 的 XML 文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    Android:id="@+id/mylayout"
    Android:layout_width="fill_parent"
    Android:layout_height="fill_parent"
    Android:orientation="vertical"
    xmlns:Android="http://schemas.Android.com/apk/res/Android">
<TextView
    Android:id="@+id/mytext1"
    Android:layout_width="165dp"
    Android:layout_height="38dp"
    Android:text="mytext1" />
</LinearLayout>
```

运行结果界面如图4.20所示。



图4.20 模拟事件响应

4.4.5 菜单事件

Android的菜单正常情况下都是隐藏的,其主要目的是节省空间,而要调用的时候单击menu按钮就弹出来。编写菜单事件一般情况下分成两步:①创建和初始化菜单;②菜单项事件处理。

Activity类有一些与菜单相关的事件方法,这些方法的定义如下:

(1) public boolean onPrepareOptionsMenu(Menu menu);——在显示选项菜单之前调用。一般用来修改即将显示的选项菜单。

(2) public void onOptionsMenuClosed(Menu menu);——在关闭选项菜单时被调用。

(3) public void onContextMenuClosed(Menu menu);——在关闭上下文菜单时被调用。

(4) public boolean onMenuOpened(int featureId,Menu menu);——在显示选项菜单之前被调用。该方法在onPrepareOptionsMenu方法之后调用。

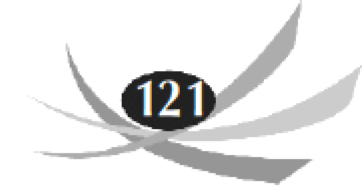
以下代码展示菜单事件,Activity程序如下:

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.Menu;
import Android.view.MenuItem;
import Android.widget.TextView;
public class EventHActivity extends Activity {
    TextView m_eventshow;
    private final int first=Menu.FIRST;
    private final int second=Menu.FIRST+1;
    private final int third=Menu.FIRST+2;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.event_8);
        setTitle("MenuEvent ");
        m_eventshow= (TextView)this.findViewById(R.id.mytext1);
        m_eventshow.setText("初始状态 start");
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        //第一个 int类型的 group ID参数,代表的是组概念,可以将几个菜单项归为一组,
        //以便更好地以组的方式管理菜单按钮
        //第二个 int类型的 item ID参数,代表的是项目编号
        //这个参数非常重要,一个 item ID对应一个 menu中的选项。在后面使用菜单的时候,
        //就靠这个 item ID来判断使用的是哪个选项
        //第三个 int类型的 order ID参数,代表的是菜单项的显示顺序。默认是 0,表示菜单
        //的显示顺序就是按照 add的显示顺序来显示
        //第四个 String类型的 title参数,表示选项中显示的文字
        menu.add(0, first, 0, "第一个 first");
        menu.add(0, second, 0, "第二个 second");
        menu.add(0, third, 0, "第三个 third");
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()) {
        case first:
            m_eventshow.setText("这是第一个子菜单!!!!");
            return true;
        case second:
            m_eventshow.setText("这是第二个子菜单!!!!");
            return true;
        case third:
            m_eventshow.setText("这是第三个子菜单!!!!");
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

界面布局的 XML 文件如下：

```
<?xml version= "1.0" encoding= "utf-8"?>
<LinearLayout
    Android:id= "@+ id/mylayout"
    Android:layout_width= "fill_parent"
    Android:layout_height= "fill_parent"
```




```
        Android:orientation= "vertical"
        xmlns:Android= "http://schemas.Android.com/apk/res/Android">
    <TextView
        Android:id= "@+id/mytext1"
        Android:layout_width= "165dp"
        Android:layout_height= "38dp"
        Android:text= "mytext1" />
    </LinearLayout>
```

处理过程界面如下。

开始界面如图 4.21 所示,单击第一个菜单如图 4.22 所示,单击第二个菜单如图 4.23 所示。

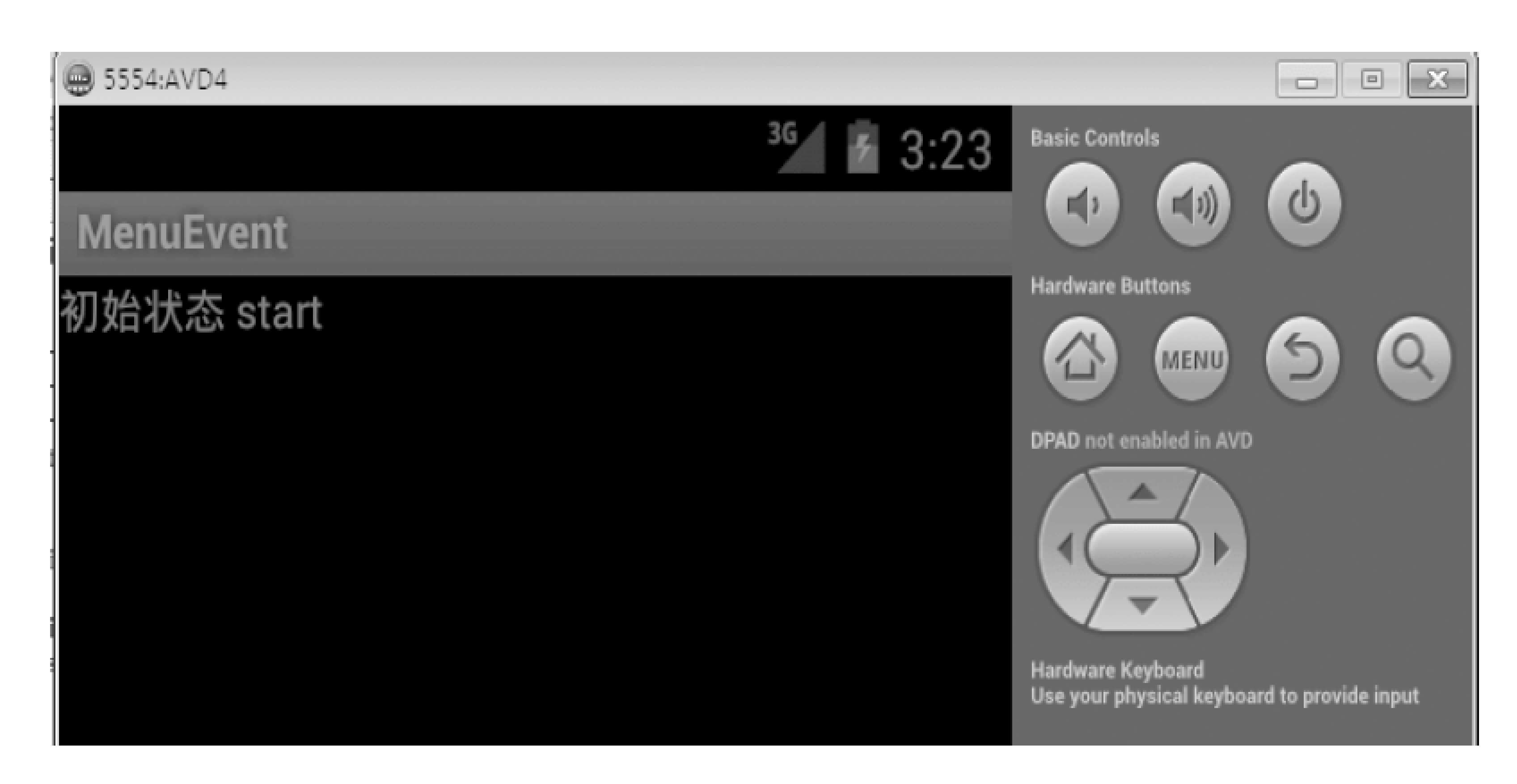


图 4.21 起始界面

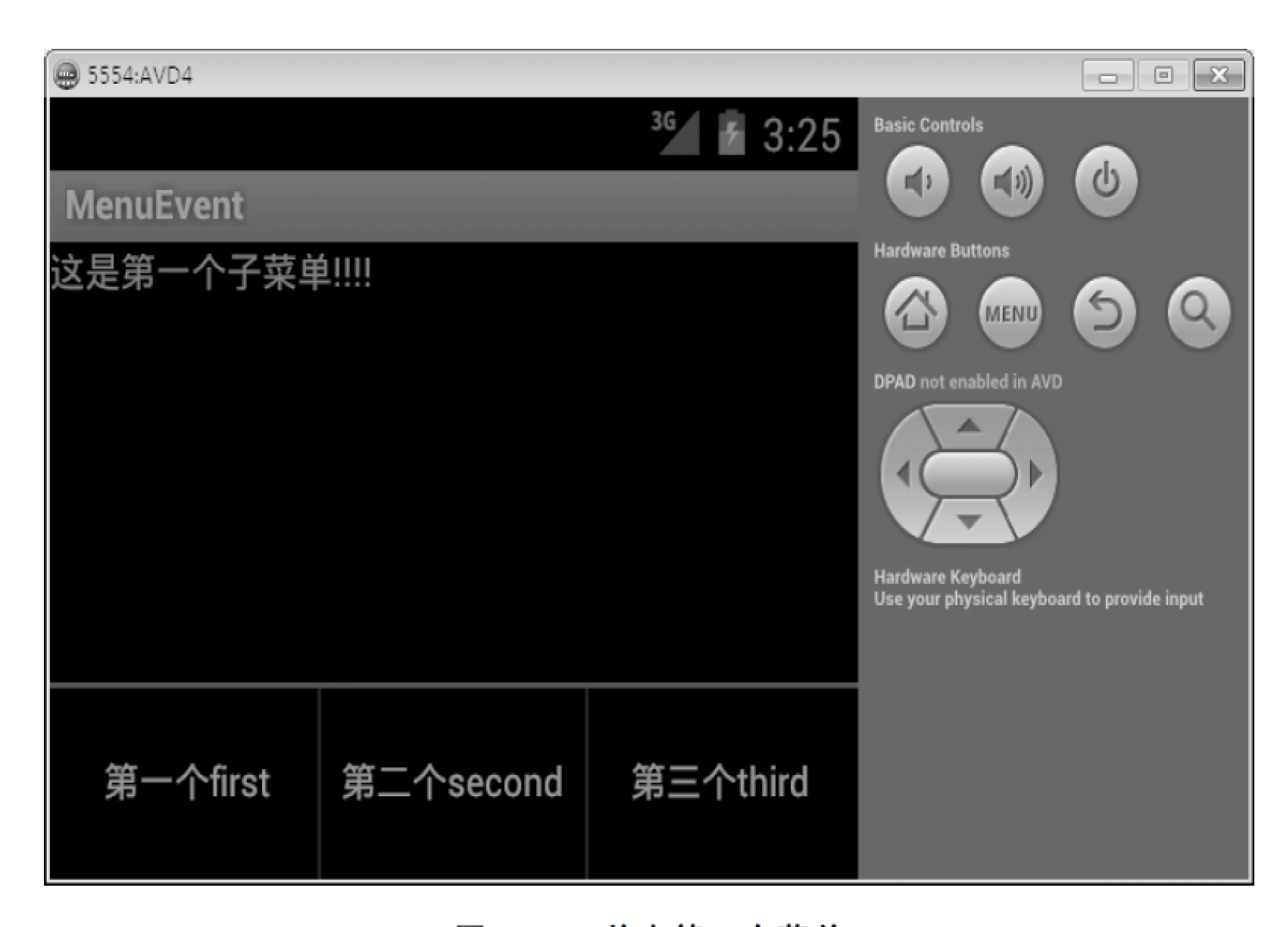


图 4.22 单击第一个菜单



图 4.23　单击第二个菜单

第 5 章 Android 布局

通过前两章的学习，我们已经可以实现简单的 Android 界面，本章将学习界面的布局，帮助我们实现更美观的界面。

先看一下如图 5.1 所示界面。

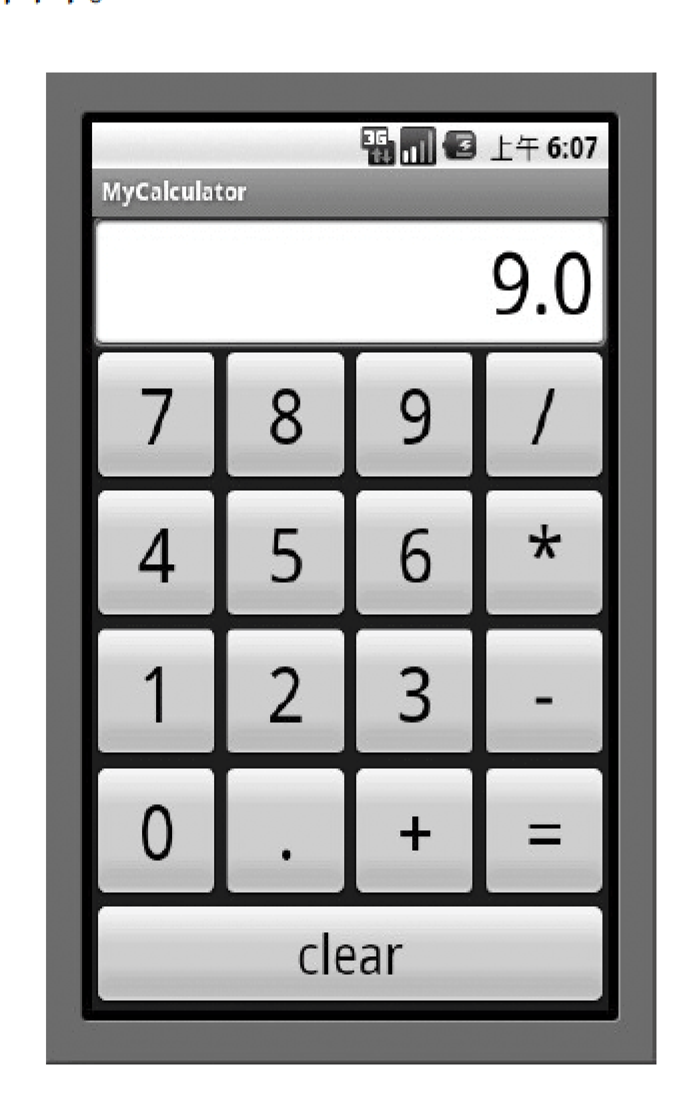


图 5.1　计算器

如果仅使用前 4 章的知识去实现这个界面，非常困难。在学习了本章之后，实现就比较简单了。5.9 节会提供计算器的界面代码供大家参考。

5.1 布局概述

在一个 Android 应用程序中，用户界面通过 View 和 ViewGroup 对象构建。Android 中有很多种 View 和 ViewGroup，它们都继承自 View 类。View 对象是 Android 平台上表示用户界面的基本单元。View 的布局显示方式直接影响用户界面，View 的布局方式是指一组 View 元素如何布局，准确地说是一个 ViewGroup 中包含的一些 View 如何布局。ViewGroup 类是布局(Layout)和视图容器(View Container)的基类。

ViewGroup 子类结构如图 5.2 所示。

Android 有七大布局对象，分别是 FrameLayout(框架布局)、LinearLayout(线性布局)、AbsoluteLayout(绝对布局)、RelativeLayout(相对布局)、TableLayout(表格布局)、GridView(网格视图)和 ListView(列表视图)。其他的一些布局都是从这 7 个布局扩展而来的。

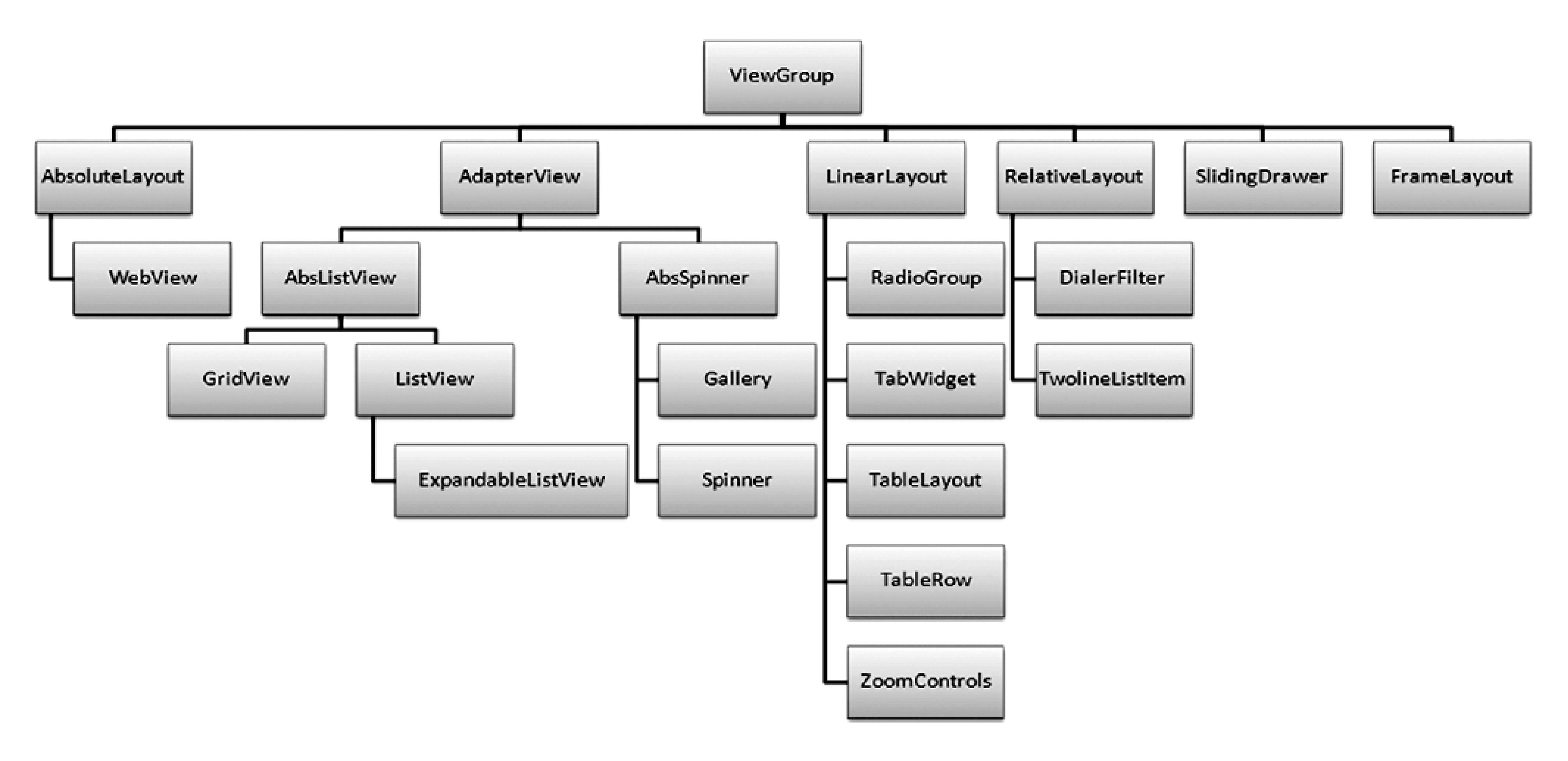


图 5.2 ViewGroup 结构图

5.2 LinearLayout

LinearLayout(线性布局)是一个 ViewGroup 以线性方向显示它的子视图元素,即垂直地或水平地,还可以嵌套,此布局运用的非常多。

常用属性如表 5-1 所示。

举例说明,看图 5.3 所示的界面。

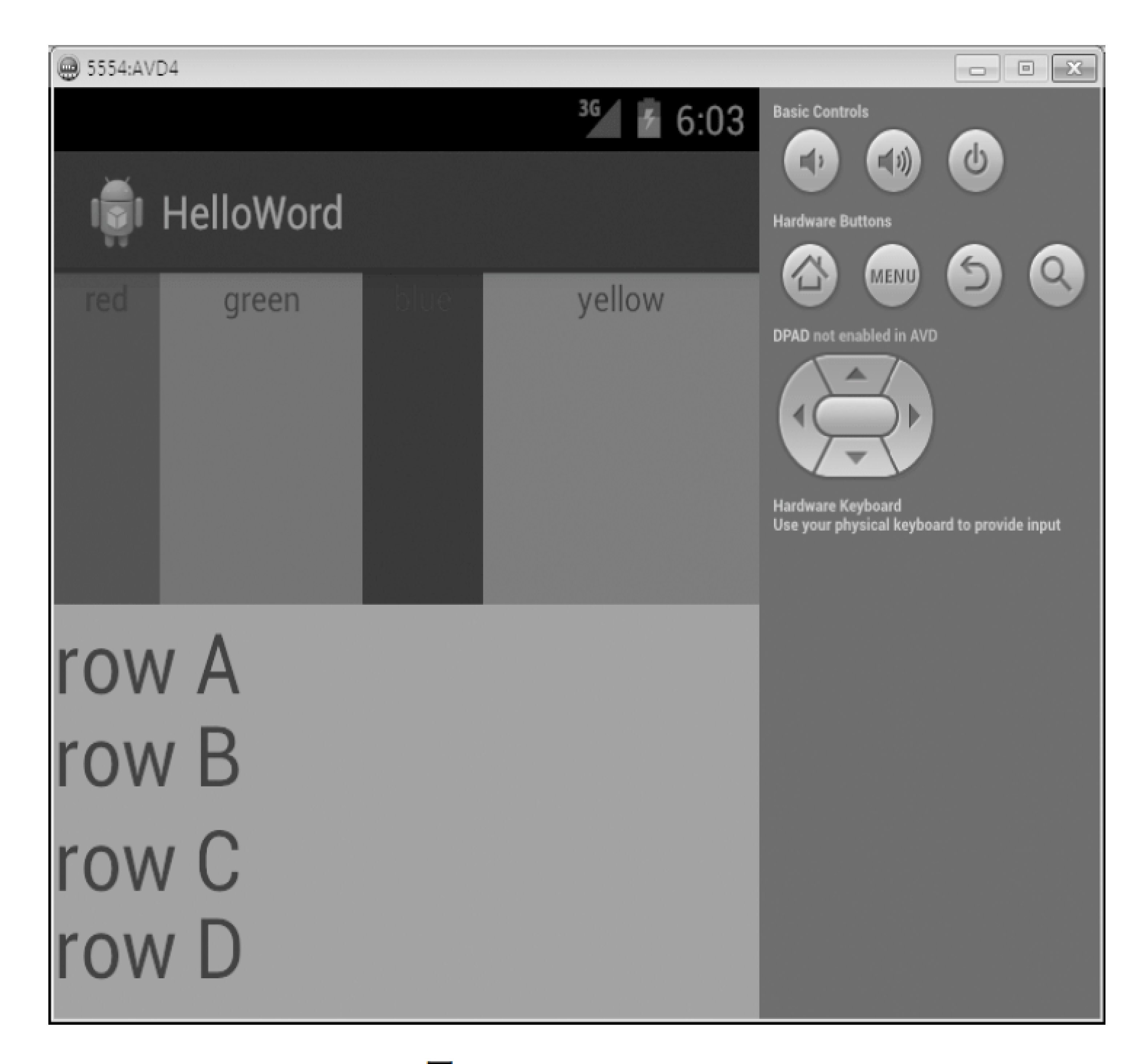


图 5.3 LinearLayout

表 5-1 LinearLayout 属性

属性名	关联方法	描　述	取　值
Android：baselineAligned	setBaselineAligned(boolean)	当设置为 false，防止布局调整子视图的基线。当子视图有不同权重时，这个属性特别有用，默认值是 true	是布尔值，"true"或"false"
Android：baselineAligned-ChildIndex	setBaselineAligned-ChildIndex(int)	当一个线性布局是基线对齐的另一个布局的一部分时，可以指定和这一布局的子女控件基线对齐	是整数值，比如"100"
Android：gravity	setGravity(int)	指定在对象内部，横纵方向上如何放置对象的内容	指定是下列常量中的一个或多个(由'|'分割)。top：将对象放在其容器的顶部，不改变其大小。bottom：将对象放在其容器的底部，不改变其大小。left：将对象放在其容器的左侧，不改变其大小。right：将对象放在其容器的右侧，不改变其大小。center_vertical：将对象纵向居中，不改变其大小。fill_vertical：必要的时候增加对象的纵向大小，以完全充满其容器。center_horizontal：将对象横向居中，不改变其大小。fill_horizontal：必要的时候增加对象的横向大小，以完全充满其容器。center：将对象横纵居中，不改变其大小。fil：必要的时候增加对象的横纵向大小，以完全充满其容器。clip_vertical：附加选项，用于按照容器的边来剪切对象的顶部和/或底部的内容。剪切基于其纵向对齐设置：顶部对齐时，剪切底部；底部对齐时剪切顶部；除此之外剪切顶部和底部。clip_horizontal：附加选项，用于按照容器的边来剪切对象的左侧和/或右侧的内容。剪切基于其横向对齐设置：左侧对齐时，剪切右侧；右侧对齐时剪切左侧；除此之外剪切左侧和右侧
Android：orientation	setOrientation(int)	行布局还是列布局	是下列常量之一：horizontal、vertical

该界面对应的 Layout 的 XML 文件如下：

```
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
Android:orientation="vertical" Android:layout_width="fill_parent"
```

```
Android:layout_height="fill_parent">
<LinearLayout Android:orientation="horizontal"
Android:layout_width="fill_parent" Android:layout_height="fill_parent"
Android:layout_weight="1">
  <TextView Android:text="red" Android:gravity="center_horizontal"
  Android:background="#aa0000" Android:layout_width="wrap_content"
  Android:layout_height="fill_parent" Android:layout_weight="1" />
  <TextView Android:text="green" Android:gravity="center_horizontal"
  Android:background="#00aa00" Android:layout_width="wrap_content"
  Android:layout_height="fill_parent" Android:layout_weight="2" />
  <TextView Android:text="blue" Android:gravity="center_horizontal"
  Android:background="#0000aa" Android:layout_width="wrap_content"
  Android:layout_height="fill_parent" Android:layout_weight="1" />
  <TextView Android:text="yellow" Android:gravity="center_horizontal"
  Android:background="#aaaa00" Android:layout_width="wrap_content"
  Android:layout_height="fill_parent" Android:layout_weight="3" />
</LinearLayout>
<LinearLayout Android:orientation="vertical"
Android:layout_width="fill_parent" Android:layout_height="fill_parent"
Android:layout_weight="1">
  <TextView Android:text="row A" Android:textSize="15pt"
  Android:layout_width="fill_parent" Android:layout_height="wrap_content"
  Android:layout_weight="1" />
<TextView Android:text="row B" Android:textSize="15pt"
  Android:layout_width="fill_parent" Android:layout_height="wrap_content"
  Android:layout_weight="2" />
  <TextView Android:text="row C" Android:textSize="15pt"
  Android:layout_width="fill_parent" Android:layout_height="wrap_content"
  Android:layout_weight="4" />
  <TextView Android:text="row D" Android:textSize="15pt"
  Android:layout_width="fill_parent" Android:layout_height="wrap_content"
  Android:layout_weight="1" />
</LinearLayout>
</LinearLayout>
```

对应的 Activity 代码，大家可以根据前面学习的内容自己编写。

需要特别说明的是 layout_weight 这个属性。它用于给一个线性布局中的诸多视图的重要程度赋值。所有的视图都有一个 layout_weight 值，默认为零，意思是需要显示多大的视图就占据多大的屏幕空间。若赋一个高于零的值，则将父视图中的可用空间分割，分割大小具体取决于每一个视图的 layout_weight 值以及该值在当前屏幕布局的整体 layout_weight 值和在其他视图屏幕布局的 layout_weight 值中所占的比率。举个例子，我们在水平方向上有一个文本标签和两个文本编辑元素。该文本标签并无指定 layout_weight 值，所以它将占据需要提供的最少空间。如果两个文本编辑元素每一个的 layout_

weight 值都设置为 1,则二者平分在父视图布局剩余的宽度(因为我们声明这二者的重要度相等)。如果两个文本编辑元素的第一个的 layout_weight 值设置为 1,而第二个的设置为 2,则第二个占据的宽度就应该是第一个的两倍。如图 5.3 所示,横向的 LinearLayout 中第一个 layout_weight 值为 1,第二个值为 2,第三个值为 1,第四个值为 3。在界面上就可以看到第一个和第三个宽度一样,第二个大一些,第四个又大一些。

5.3　RelativeLayout

RelativeLayout(相对布局)是一个 ViewGroup 以相对位置显示它的子视图元素,一个视图可以指定相对于它的兄弟视图的位置(例如在给定视图的左边或者下面)或相对于 RelativeLayout 的特定区域的位置(例如底部对齐,或中间偏左)。

相对布局是设计用户界面的有力工具,因为它消除了嵌套视图组。如果你发现使用了多个嵌套的 LinearLayout 视图组后,可以考虑使用一个 RelativeLayout 视图组了。

RelativeLayout 允许子元素指定他们相对于其他元素或父元素的位置(通过 ID 指定)。因此,可以以右对齐,或上下,或置于屏幕中央的形式来排列两个元素。元素按顺序排列,因此如果第一个元素在屏幕的中央,那么相对于这个元素的其他元素将以屏幕中央的相对位置来排列。

常用属性如表 5-2 所示。

表 5-2　RelativeLayout 属性

属　性　名	关联方法	描　　述	取　　值
Android:gravity	setGravity(int)	指定在对象内部,横纵方向上如何放置对象的内容	与 LinearLayout 相同
Android:ignoreGravity	setIgnoreGravity(int)	指定不受对齐方式影响的视图	是对其他资源的参照,形式为"@[+][package:]type:name"或"?[package:][type:]name"形式的主题属性
Android:layout_above		将该控件的底部置于给定 ID 控件之上	同上
Android:layout_below		将该控件的顶部置于给定 ID 控件之下	同上
Android:layout_toLeftOf		将该控件的右边缘和给定 ID 控件的左边缘对齐	同上
Android:layout_toRightOf		将该控件的左边缘和给定 ID 控件的右边缘对齐	同上
Android:layout_alignBaseline		将该控件的 baseline 和给定 ID 控件的 baseline 对齐	同上
Android:layout_alignBottom		将该控件的底部边缘与给定 ID 控件的底部边缘对齐	同上

续表

属　性　名	关联方法	描　　述	取　　值
Android:layout_alignLeft		将该控件的左边缘与给定ID控件的左边缘对齐	同上
Android:layout_alignRight		将该控件的右边缘与给定ID控件的右边缘对齐	同上
Android:layout_alignTop		将该控件的顶部边缘与给定ID控件的顶部对齐	同上
Android:alignParentBottom		如果该值为true,则将该控件的底部和父控件的底部对齐	布尔值:true或false
Android:layout_alignParentLeft		如果该值为true,则将该控件左边与父控件的左边对齐	同上
Android:layout_alignParentRight		如果该值为true,则将该控件的右边与父控件的右边对齐	同上
Android:layout_alignParentTop		如果该值为true,则将该控件的顶部与父控件的顶部对齐	同上
Android:layout_centerHorizontal		如果为真,该控件将被置于水平方向的中央	同上
Android:layout_centerInParent		如果为真,该控件将被置于父控件水平方向和垂直方向的中央	同上
Android:layout_centerVertical		如果为真,该控件将被置于父控件水平方向和垂直方向的中央	同上
Android:layout_marginLeft		此属性用来设置控件之间的间隙(控件和控件之间和内边距不同)	尺寸值
Android:padding="3dip"		说明了四边的内边距是3dip	尺寸值

举例说明,界面如图5.4所示。



图5.4　RelativeLayout

该界面布局的 XML 文件如下：

```
<RelativeLayout
    xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:layout_width="fill_parent" Android:layout_height="wrap_content"
    Android:background="@drawable/blue" Android:padding="10dip">
    <TextView Android:id="@+id/label" Android:layout_width="fill_parent"
        Android:layout_height="wrap_content" Android:text="请输入邮箱:"/>
    <EditText Android:id="@+id/email" Android:layout_width="fill_parent"
        Android:layout_height="wrap_content"
        Android:background="@Android:drawable/editbox_background"
        Android:layout_below="@id/label"/>
    <Button Android:id="@+id/cancel" Android:layout_width="wrap_content"
        Android:layout_height="wrap_content" Android:layout_below="@id/email"
        Android:layout_alignParentRight="true"
        Android:layout_marginLeft="10dip" Android:text="取消"/>
    <Button Android:id="@+id/ok" Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_toLeftOf="@id/cancel"
        Android:layout_alignTop="@id/cancel" Android:text="确定"/>
</RelativeLayout>
```

对应的 Activity 代码，大家可以根据前面学习的内容自己编写。

5.4　TableLayout

TableLayout（表格布局）是一个 ViewGroup 以表格显示它的子视图元素，即行和列标识一个视图的位置。其实 Android 的表格布局跟 HTML 中的表格布局非常类似，TableRow 就像 HTML 表格的<tr>标记。

用表格布局需要知道以下几点：

（1）Android：shrinkColumns，对应的方法：setShrinkAllColumns（boolean），作用：设置表格的列是否收缩（列编号从 0 开始，下同），多列用逗号隔开（下同），如 Android：shrinkColumns="0，1，2"，即表格的第 1、2、3 列的内容是收缩的以适合屏幕，不会挤出屏幕。

（2）Android：collapseColumns，对应的方法：setColumnCollapsed（int，boolean），作用：设置表格的列是否隐藏。

（3）Android：stretchColumns，对应的方法：setStretchAllColumns（boolean），作用：设置表格的列是否拉伸。

举例说明，界面如图 5.5 和图 5.6 所示。

图 5.5 界面的布局 XML 文件如下：

图 5.5 TableLayout(1)

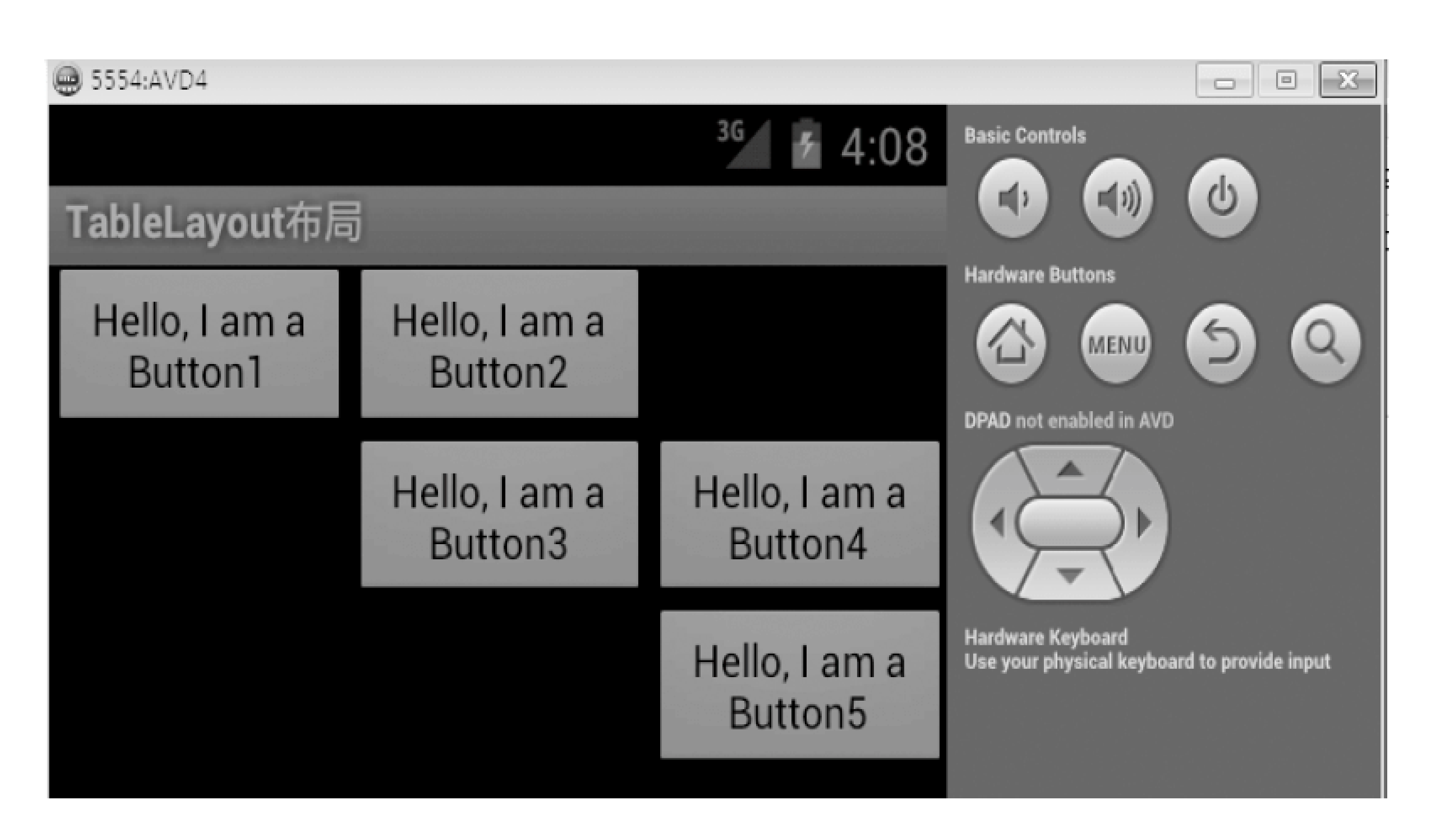


图 5.6 TableLayout(2)

```
<TableLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:layout_width="fill_parent" Android:layout_height="fill_parent"
    Android:stretchColumns="1">
    <TableRow>
        <TextView Android:text="用户 ID:" Android:textStyle="bold"
            Android:gravity="right" Android:padding="3dip" />
        <EditText Android:id="@+id/userid" Android:padding="3dip"
            Android:scrollHorizontally="true" />
    </TableRow>
    <TableRow>
        <TextView Android:text="登录密码:" Android:textStyle="bold"
            Android:gravity="right" Android:padding="3dip" />
        <EditText Android:id="@+id/password" Android:password="true"
            Android:padding="3dip" Android:scrollHorizontally="true" />
    </TableRow>
    <TableRow Android:gravity="right">
```

```
        <Button Android:id="@+id/cancel"
            Android:text="Cancel" />
        <Button Android:id="@+id/login"
            Android:text="Login" />
    </TableRow>
</TableLayout>
```

图 5.6 界面布局的 XML 文件如下：

```
<TableLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
             Android:layout_width="fill_parent"
             Android:layout_height="fill_parent"
             Android:shrinkColumns="0,1,2"><!-- have an eye on ! -->
    <TableRow><!-- row1 -->
    <Button Android:id="@+id/button1"
            Android:layout_width="wrap_content"
            Android:layout_height="wrap_content"
            Android:text="Hello, I am a Button1"
            Android:layout_column="0"/>
      <Button Android:id="@+id/button2"
     Android:layout_width="wrap_content"
     Android:layout_height="wrap_content"
     Android:text="Hello, I am a Button2"
     Android:layout_column="1" />
     </TableRow>
    <TableRow><!-- row2 -->
    <Button Android:id="@+id/button3"
            Android:layout_width="wrap_content"
            Android:layout_height="wrap_content"
            Android:text="Hello, I am a Button3"
            Android:layout_column="1" />
    <Button Android:id="@+id/button4"
          Android:layout_width="wrap_content"
          Android:layout_height="wrap_content"
          Android:text="Hello, I am a Button4"
          Android:layout_column="1"   />
    </TableRow>
    <TableRow>
          <Button Android:id="@+id/button5"
          Android:layout_width="wrap_content"
          Android:layout_height="wrap_content"
          Android:text="Hello, I am a Button5"
          Android:layout_column="2"  />
     </TableRow>
</TableLayout>
```

对应的 Activity 代码，大家可以根据前面学习的内容自己编写。

5.5 AbsoluteLayout

AbsoluteLayout(绝对布局)可以让子元素指定准确的 x、y 坐标值，并显示在屏幕上。(0，0)为左上角，当向下或向右移动时，坐标值将变大。AbsoluteLayout 没有页边框，允许元素之间互相重叠(尽管不推荐)。我们通常不推荐使用 AbsoluteLayout，除非有正当理由要使用它，因为它使界面代码太过刚性，以至于在不同的设备上可能不能很好地工作。属性设置如表 5-3 所示。

表 5-3 AbsoluteLayout 属性

属　性	功　能	取　值
Android:layout_x	设置按钮的 x 坐标	尺寸值
Android:layout_y	设置按钮的 y 坐标	尺寸值
Android:layout_width	设置按钮的宽度	尺寸值

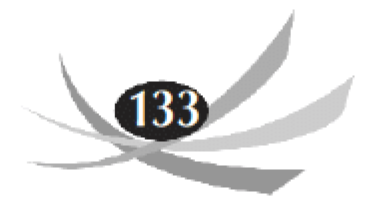
举例说明，界面如图 5.7 所示。



图 5.7 AbsoluteLayout

该界面布局的 XML 文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:orientation="vertical"
        Android:layout_width="fill_parent"
        Android:layout_height="fill_parent" >
    <EditText
        Android:text="Welcome to input"
        Android:layout_width="fill_parent"
        Android:layout_height="wrap_content" />
    <Button
```



```
            Android:layout_width="wrap_content"
            Android:layout_height="wrap_content"
            Android:layout_x="240dp"
            Android:layout_y="57dp"
            Android:text="确认" />
    </AbsoluteLayout>
```

对应的 Activity 代码,大家可以根据前面学习的内容自己编写。

5.6 FrameLayout

FrameLayout(框架布局)是最简单的一个布局对象。所有显示对象都将固定在屏幕的左上角,不能指定位置,但允许有多个显示对象,只是后一个会直接覆盖在前一个之上显示,会把前面的组件部分或全部挡住。

举例说明,界面如图 5.8 所示。

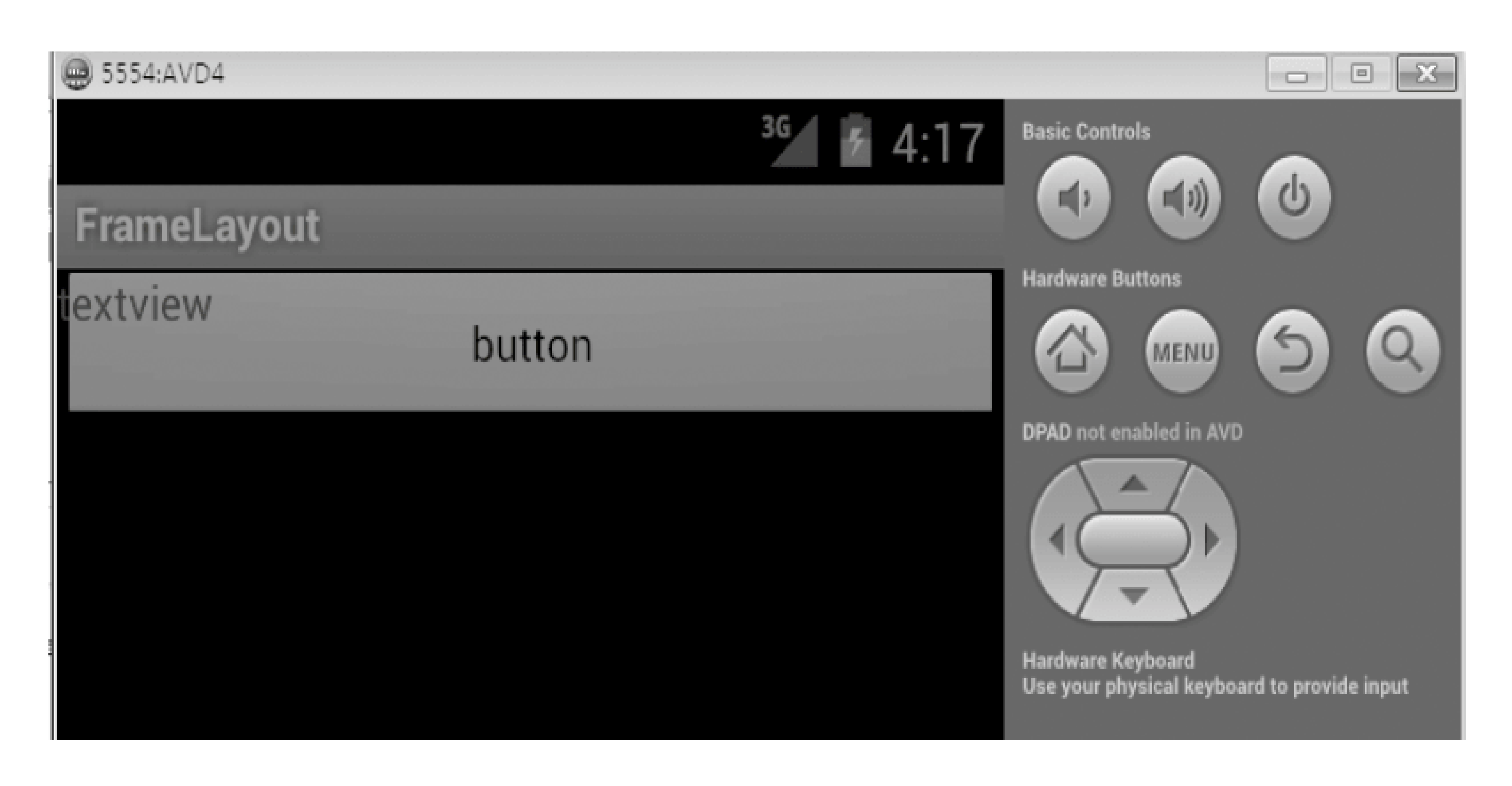


图 5.8 FrameLayout

该界面布局的 XML 文件如下:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
   Android:layout_width="fill_parent"
    Android:layout_height="fill_parent" >
<Button
    Android:text="button"
    Android:layout_width="fill_parent"
    Android:layout_height="wrap_content"    />
<TextView
    Android:text="textview"
    Android:textColor="#0000ff"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content" />
</FrameLayout>
```

对应的 Activity 代码,大家可以根据前面学习的内容自己编写。

5.7 GridView

GridView(网格视图)控件用于把一系列的空间组织成一个二维的网格显示出来。应用的比较多的就是组合图片显示。GridView 和 ListView 都是比较常用的多控件布局,而 GridView 更是实现九宫图的首选。属性设置如表 5-4 所示。

表 5-4 GridView 属性

属性名	关联方法	描述
Android:columnWidth	setColumnWidth(int)	设置列的宽度
Android:gravity	setGravity(int)	指定在对象内部、横纵方向上如何放置对象的内容
Android:horizontalSpacing	setHorizontalSpacing(int)	两列之间的间距
Android:numColumns	setNumColumns(int)	列数
Android:stretchMode	setStretchMode(int)	缩放模式
Android:verticalSpacing	setVerticalSpacing(int)	两行之间的间距

举例说明,界面如图 5.9 所示。

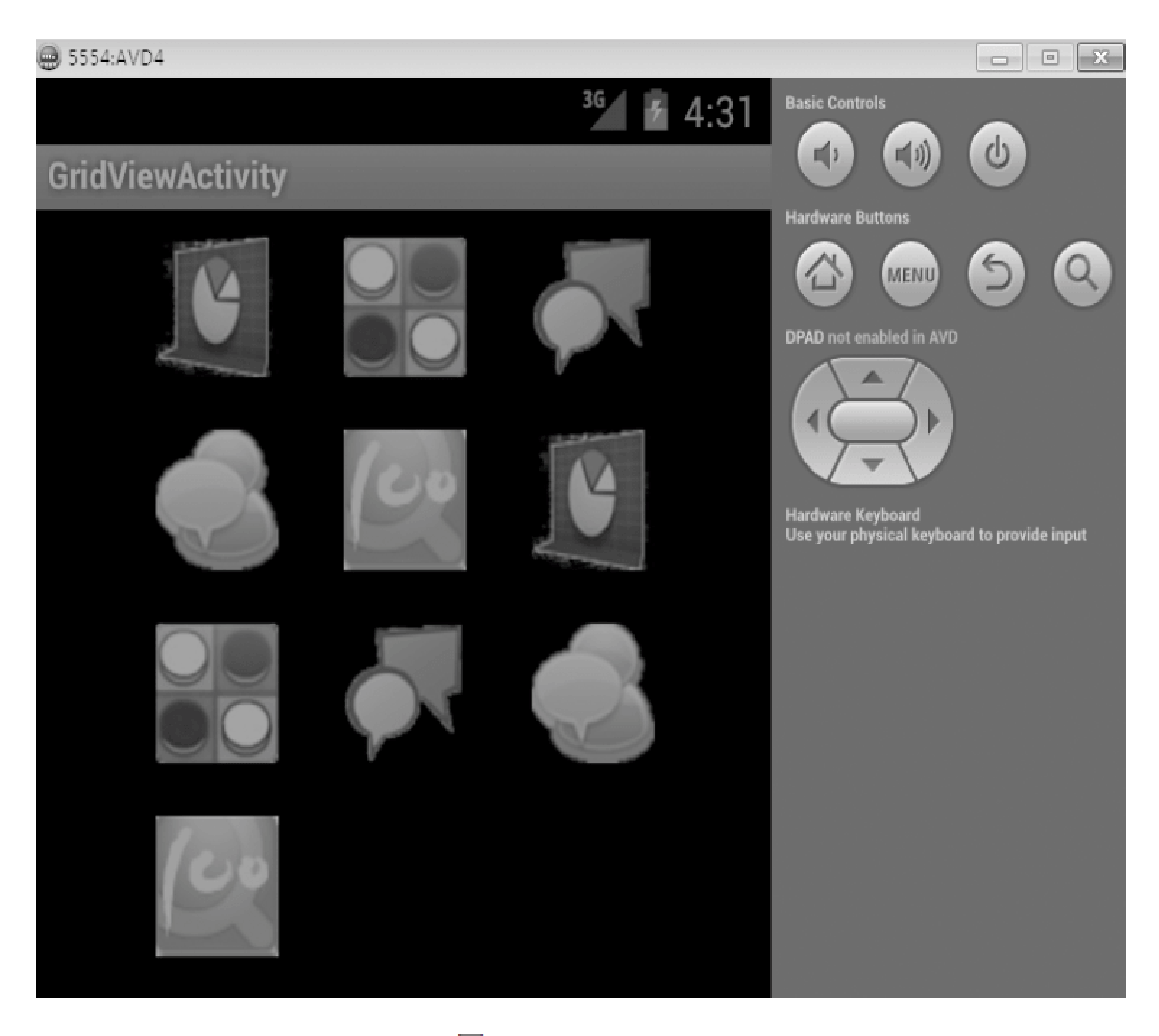


图 5.9 GridView

该界面布局的 XML 文件如下:

```
<GridView xmlns:Android="http://schemas.Android.com/apk/res/Android"
```

```
    Android:id= "@ + id/grid_view"
    Android:layout_width= "fill_parent"
    Android:layout_height= "fill_parent"
    Android:numColumns= "auto_fit"
    Android:verticalSpacing= "10dp"
    Android:horizontalSpacing= "10dp"
    Android:columnWidth= "90dp"
    Android:stretchMode= "columnWidth"
    Android:gravity= "center"
/>
```

该界面的对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.content.Context;
import Android.os.Bundle;
import Android.view.View;
import Android.view.ViewGroup;
import Android.widget.BaseAdapter;
import Android.widget.GridView;
import Android.widget.ImageView;
public class GridViewActivity   extends Activity   {
    private Integer[] mThumbIds= {
            R.drawable.image_thumb_0, R.drawable.image_thumb_1,
            R.drawable.image_thumb_2, R.drawable.image_thumb_3,
            R.drawable.image_thumb_4, R.drawable.image_0,
            R.drawable.image_1, R.drawable.image_2,
            R.drawable.image_3, R.drawable.image_4
    };
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.grid_view);
        setTitle("GridViewActivity");
        GridView gridview= (GridView) findViewById(R.id.grid_view);
        gridview.setAdapter(new ImageAdapter(this));
    }
    public class ImageAdapter extends BaseAdapter {
        private Context mContext;
        public ImageAdapter(Context c) {
            mContext= c;
        }
        public int getCount() {
            return mThumbIds.length;
        }
```

```
            public Object getItem(int position) {
                return null;
            }
            public long getItemId(int position) {
                return 0;
            }
            public View getView(int position, View convertView,
                                          ViewGroup parent) {
                ImageView imageView;
                if(convertView= = null) {
                        //if it's not recycled, initialize some attributes
                    imageView= new ImageView(mContext);
                    imageView.setLayoutParams(new GridView.LayoutParams(85,
                                                                85));
                    imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
                    imageView.setPadding(8, 8, 8, 8);
                } else {
                    imageView= (ImageView) convertView;
                }
                imageView.setImageResource(mThumbIds[position]);
                return imageView;
            }
        }
    }
```

5.8 ListView

ListView(列表视图)是比较常用的组件,它以列表的形式展示具体内容,并且能够根据数据的长度自适应显示。属性设置如表 5-5 所示。

表 5-5 ListView 属性

属 性 名	描 述	取 值
Android:choiceMode	为列表视图设置选择时的行为。默认列表被选择时没有任何行为。设置选择模式为单选时,列表中只能有一个条目处于选择状态。设置为多选时,列表允许选择多个条目	是下列常量之一。none:普通列表不显示选择状态;singleChoice:列表允许选择一个条目;multipleChoice:列表允许选择多个条目
Android:divider	在列表条目之间显示的可绘制对象或颜色	可能是对其他资源的参照,形式为"@[+][package:]type:name"或"?[package:][type:]name"的主题属性。可能是颜色值

续表

属 性 名	描 述	取 值
Android:dividerHeight	分隔符的高度。如果未指定,使用分隔符的固有高度	是尺寸值
Android:entries	对用于在列表视图中显示的数组资源的引用。对于静态对象,这样赋值比用程序处理要简单	是尺寸值
Android:footerDividersEnabled	设为假时,列表视图在表尾视图间不绘制分隔符。默认值为真	是布尔值
Android:headerDividersEnabled	设为假时,列表视图在表头视图间不绘制分隔符。默认值为真	是布尔值

举例说明,界面如图5.10所示。

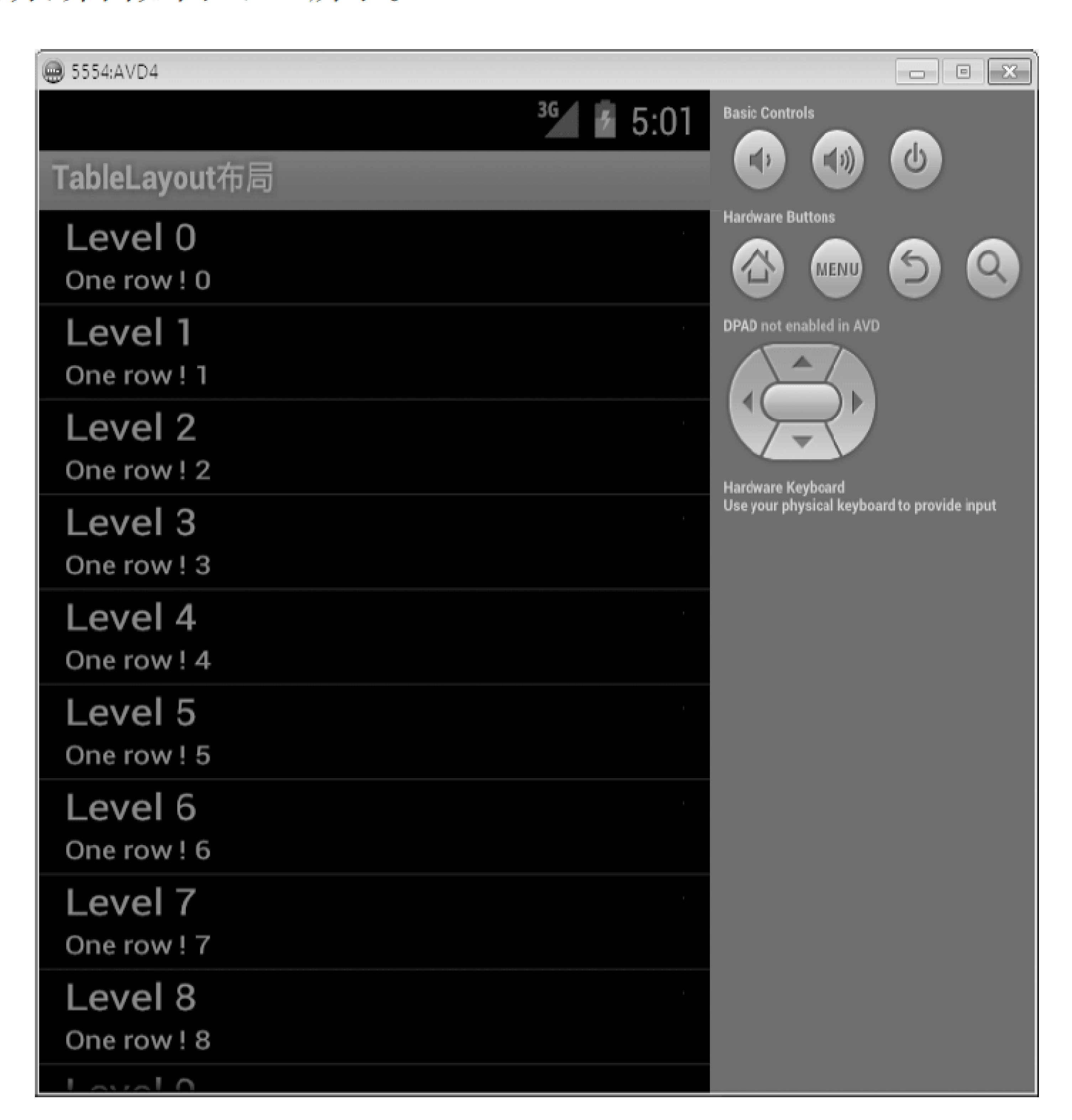


图5.10 ListView

该界面对应的ListView布局的XML文件如下:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    Android:id="@+id/LinearLayout01"
    Android:layout_width="fill_parent"
    Android:layout_height="fill_parent"
    xmlns:Android="http://schemas.Android.com/apk/res/Android">
    <ListView Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
```

```
        Android:id="@+id/ListView01"/>
</LinearLayout>
```

该界面对应的ListView的item布局的XML文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    Android:id="@+id/RelativeLayout01"
    Android:layout_width="fill_parent"
    xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:layout_height="wrap_content"
    Android:paddingBottom="4dip"
    Android:paddingLeft="12dip"
    Android:paddingRight="12dip">
<ImageView
    Android:paddingTop="12dip"
    Android:layout_alignParentRight="true"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:id="@+id/ItemImage"/>
<TextView
    Android:text="TextView01"
    Android:layout_height="wrap_content"
    Android:textSize="20dip"
    Android:layout_width="fill_parent"
    Android:id="@+id/ItemTitle"/>
<TextView
    Android:text="TextView02"
    Android:layout_height="wrap_content"
    Android:layout_width="fill_parent"
    Android:layout_below="@+id/ItemTitle"
    Android:id="@+id/ItemText"/>
</RelativeLayout>
```

对应的Activity代码如下：

```
import Java.util.ArrayList;
import Java.util.HashMap;
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.ContextMenu;
import Android.view.MenuItem;
import Android.view.View;
import Android.view.ContextMenu.ContextMenuInfo;
import Android.view.View.OnCreateContextMenuListener;
import Android.widget.AdapterView;
```

```
import Android.widget.ListView;
import Android.widget.SimpleAdapter;
import Android.widget.AdapterView.OnItemClickListener;
public class ListViewActivity extends Activity {
    ListView list;
    @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.listview_layout);
            //绑定 Layout里面的 ListView
            ListView list= (ListView) findViewById(R.id.ListView01);
            //生成动态数组,加入数据
            ArrayList<HashMap<String, Object>> listItem=
                    new ArrayList<HashMap<String, Object>>();
            for(int i=0;i<10;i++)
            {
                HashMap<String, Object> map= new HashMap<String, Object>();
                map.put("ItemImage", R.drawable.blue);          //图像资源的 ID
                map.put("ItemTitle", "Level "+ i);
                map.put("ItemText", "One row ! "+ i);
                listItem.add(map);
            }
            //生成适配器的 Item和动态数组对应的元素
            SimpleAdapter listItemAdapter=
                        new SimpleAdapter(this, listItem,       //数据源
                R.layout.list_items,//ListItem的 XML实现
                //动态数组与 ImageItem对应的子项
                new String[] {"ItemImage","ItemTitle", "ItemText"},
                //ImageItem的 XML文件里面的一个 ImageView,两个 TextView ID
                new int[] {R.id.ItemImage,R.id.ItemTitle,R.id.ItemText}
            );
            //添加并且显示
            list.setAdapter(listItemAdapter);
            //添加单击
            list.setOnItemClickListener(new OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> arg0, View arg1,
                                                        int arg2,
                        long arg3) {
                    setTitle("单击第"+ arg2+ "个项目");
                }
            });
        //添加长按单击
            list.setOnCreateContextMenuListener(
```

```
                        new OnCreateContextMenuListener() {
                    @Override
                    public void onCreateContextMenu(ContextMenu menu,
                                    View v,ContextMenuInfo menuInfo) {
                        menu.setHeaderTitle("长按菜单-ContextMenu");
                        menu.add(0, 0, 0, "弹出长按菜单 A");
                        menu.add(0, 1, 0, "弹出长按菜单 B");
                    }
                });
            }
            //长按菜单响应函数
            @Override
            public boolean onContextItemSelected(MenuItem item) {
                setTitle("单击了长按菜单里面的第"+ item.getItemId()+ "个项目");
                return super.onContextItemSelected(item);
            }
}
```

5.9 计算器的实现

界面布局的 XML 文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:layout_width="fill_parent"
    Android:layout_height="wrap_content"
    Android:textSize="42sp" Android:stretchColumns="1">
    <TableRow>
      <EditText Android:id="@+id/result"
        Android:layout_width="fill_parent"
Android:layout_height="wrap_content"
Android:background="@Android:drawable/editbox_background"
  Android:layout_span="4" Android:textSize="48sp"
  Android:gravity="right|center_vertical"
Android:cursorVisible="false"
   Android:editable="false" Android:lines="1" />
</TableRow>
<TableRow>
<LinearLayout Android:orientation="horizontal"
     Android:layout_width="fill_parent"
     Android:layout_height="wrap_content"
     Android:textSize="42sp" Android:layout_weight="1">
     <Button Android:id="@+id/num7" Android:layout_width="fill_parent"
```

```
Android:layout_height= "wrap_content" Android:textSize= "42sp"
Android:text= "7" Android:layout_weight= "1" />
<Button Android:id= "@ + id/num8" Android:layout_width= "fill_parent"                    Android:layout_height= "
wrap_content" Android:textSize= "42sp"
Android:text= "8" Android:layout_weight= "1" />
<Button Android:id= "@ + id/num9" Android:layout_width= "fill_parent"
  Android:layout_height= "wrap_content" Android:textSize= "42sp"
  Android:text= "9" Android:layout_weight= "1" />
  <Button Android:id= "@ + id/divide" Android:layout_width= "fill_parent"
          Android:layout_height= "wrap_content" Android:textSize= "42sp"
          Android:text= "/" Android:layout_weight= "1" />
    </LinearLayout>
</TableRow>
<TableRow>
    <LinearLayout Android:orientation= "horizontal"
Android:layout_width= "fill_parent" Android:layout_height= "wrap_content
Android:textSize= "42sp" Android:layout_weight= "1">
<Button Android:id= "@ + id/num4" Android:layout_width= "fill_parent"
  Android:layout_height= "wrap_content" Android:textSize= "42sp"
Android:text= "4" Android:layout_weight= "1" />
<Button Android:id= "@ + id/num5" Android:layout_width= "fill_parent"
          Android:layout_height= "wrap_content" Android:textSize= "42sp"
          Android:text= "5" Android:layout_weight= "1" />
<Button Android:id= "@ + id/num6" Android:layout_width= "fill_parent"
          Android:layout_height= "wrap_content" Android:textSize= "42sp"
        Android:text= "6" Android:layout_weight= "1" />
<Button Android:id= "@ + id/multiply" Android:layout_width= "fill_parent"
          Android:layout_height= "wrap_content" Android:textSize= "42sp"
          Android:text= " * " Android:layout_weight= "1" />
</LinearLayout>
</TableRow>
<TableRow>
<LinearLayout Android:orientation= "horizontal"
Android:layout_width= "fill_parent" Android:layout_height= "wrap_content"
Android:textSize= "42sp" Android:layout_weight= "1">
<Button Android:id= "@ + id/num1" Android:layout_width= "fill_parent"
Android:layout_height= "wrap_content" Android:textSize= "42sp"
Android:text= "1" Android:layout_weight= "1" />
<Button Android:id= "@ + id/num2" Android:layout_width= "fill_parent"
          Android:layout_height= "wrap_content" Android:textSize= "42sp"
          Android:text= "2" Android:layout_weight= "1" />
<Button Android:id= "@ + id/num3" Android:layout_width= "fill_parent"
   Android:layout_height= "wrap_content" Android:textSize= "42sp"
             Android:text= "3" Android:layout_weight= "1" />
```

```
        <Button Android:id="@+id/subtract" Android:layout_width="fill_parent"
          Android:layout_height="wrap_content" Android:textSize="42sp"
        Android:text="-" Android:layout_weight="1" />
          </LinearLayout>
    </TableRow>
    <TableRow>
    <LinearLayout Android:orientation="horizontal"
     Android:layout_width="fill_parent" Android:layout_height="wrap_content"
     Android:textSize="42sp" Android:layout_weight="1">
     <Button Android:id="@+id/num0" Android:layout_width="fill_parent"
     Android:layout_height="wrap_content" Android:textSize="42sp"
     Android:text="0" Android:layout_weight="1" />
     <Button Android:id="@+id/point" Android:layout_width="fill_parent"
     Android:layout_height="wrap_content" Android:textSize="42sp"
     Android:text="." Android:layout_weight="1" />
     <Button Android:id="@+id/add" Android:layout_width="fill_parent"
     Android:layout_height="wrap_content" Android:textSize="42sp"
     Android:text="+" Android:layout_weight="1" />
     <Button Android:id="@+id/equal" Android:layout_width="fill_parent"
     Android:layout_height="wrap_content" Android:textSize="42sp"
     Android:text="=" Android:layout_weight="1" />
     </LinearLayout>
    </TableRow>
    <TableRow>
          <Button Android:id="@+id/clear" Android:layout_width="fill_parent"
           Android:layout_height="wrap_content" Android:textSize="30sp"
          Android:text="clear" Android:layout_span="4" Android:gravity=
          "center_vertical|center_horizontal"/>
    </TableRow>
</TableLayout>
```

对应的 Activity 代码如下：

```
import Java.math.BigDecimal;
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.View;
import Android.view.View.OnClickListener;
import Android.widget.Button;
import Android.widget.EditText;
public class MyCalculator extends Activity {
    private Button[] btnNum= new Button[11];            //数值按钮
    private Button[] btnCommand= new Button[5];         //符号按钮
    private EditText editText= null;                    //显示区域
    private Button btnClear= null;                      //clear 按钮
```

```
private String lastCommand;                    //用于保存运算符
private boolean clearFlag;
  //用于判断是否清空显示区域的值,true需要,false不需要
private boolean firstFlag;
  //用于判断是否是首次输入,true首次,false不是首次
private double result;                         //计算结果
public MyCalculator() {
    //初始化各项值
    result= 0;                                 //x的值
    firstFlag= true;                           //是首次运算
    clearFlag= false;                          //不需要清空
    lastCommand= "= ";                         //运算符
}
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.calculator);
    //获取运算符
    btnCommand[0]= (Button) findViewById(R.id.add);
    btnCommand[1]= (Button) findViewById(R.id.subtract);
    btnCommand[2]= (Button) findViewById(R.id.multiply);
    btnCommand[3]= (Button) findViewById(R.id.divide);
    btnCommand[4]= (Button) findViewById(R.id.equal);
    //获取数字
    btnNum[0]= (Button) findViewById(R.id.num0);
    btnNum[1]= (Button) findViewById(R.id.num1);
    btnNum[2]= (Button) findViewById(R.id.num2);
    btnNum[3]= (Button) findViewById(R.id.num3);
    btnNum[4]= (Button) findViewById(R.id.num4);
    btnNum[5]= (Button) findViewById(R.id.num5);
    btnNum[6]= (Button) findViewById(R.id.num6);
    btnNum[7]= (Button) findViewById(R.id.num7);
    btnNum[8]= (Button) findViewById(R.id.num8);
    btnNum[9]= (Button) findViewById(R.id.num9);
    btnNum[10]= (Button) findViewById(R.id.point);
    //初始化显示结果区域
    editText= (EditText) findViewById(R.id.result);
    editText.setText("0.0");
    //实例化监听器对象
    NumberAction na= new NumberAction();
    CommandAction ca= new CommandAction();
    for (Button bc:btnCommand) {
        bc.setOnClickListener(ca);
    }
```

```
        for(Button bc:btnNum) {
            bc.setOnClickListener(na);
        }
        //clear 按钮的动作
        btnClear= (Button) findViewById(R.id.clear);
        btnClear.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View view) {
                editText.setText("0.0");
                //初始化各项值
                result= 0;                          //x 的值
                firstFlag= true;                    //是首次运算
                clearFlag= false;                   //不需要清空
                lastCommand= "=";                   //运算符
            }
        });
    }
    //数字按钮监听器
    private class NumberAction implements OnClickListener {
        @Override
        public void onClick(View view) {
            Button btn= (Button) view;
            String input= btn.getText().toString();
            if(firstFlag) {                         //首次输入
                //如果一开始直接输入".",就不处理
                if(input.equals(".")) {
                    return;
                }
                //如果是"0.0"的话,就清空
                if(editText.getText().toString().equals("0.0")) {
                    editText.setText("");
                }
                firstFlag= false;                   //改变是否首次输入的标记值
            }else{
                String editTextStr= editText.getText().toString();
                //判断显示区域的值里面是否已经有".",如果有,输入的又是".",就什么都
                //不做
                if(editTextStr.indexOf(".")!= - 1&&input.equals(".")){
                    return;
                }
                //判断显示区域的值里面是否只有"-",如果是,输入的又是".",就什么都不做
                if(editTextStr.equals("- ") && input.equals(".")){
                    return;
                }
```

```
                //判断显示区域的值是否是"0",如果是,输入的不是".",就什么也不做
                if(editTextStr.equals("0") && !input.equals(".")){
                    return;
                }
            }
            //如果单击了运算符,再输入数字,就要清空显示区域的值
            if(clearFlag){
                editText.setText("");
                clearFlag= false;                       //还原初始值,不需要清空
            }
            editText.setText(editText.getText().toString()+ input);
            //设置显示区域的值
        }
    }
    //符号按钮监听器
    private class CommandAction implements OnClickListener {
        @Override
        public void onClick(View view) {
            Button btn= (Button) view;
            String inputCommand= (String) btn.getText();
            if (firstFlag) {                            //首次输入"-"的情况
                if (inputCommand.equals("-")) {
                    editText.setText("-");              //显示区域的内容设置为"-"
                    firstFlag = false;                  //改变首次输入的标记
                }
            } else {
                if (!clearFlag) {
                          //如果 flag= false不需要清空显示区的值,就调用方法计算
                    calculate(Double.parseDouble(
                                              editText.getText().toString()));
                           //保存显示区域的值,并计算
                }
                //保存单击的运算符
                lastCommand= inputCommand;
                clearFlag= true;                        //因为这里已经输入过运算符
            }
        }
    }
    //计算用的方法
    private void calculate(double x) {
        if (lastCommand.equals("+")) {
            result += x;
        } else if(lastCommand.equals("-")) {
            result -= x;
```

```
            } else if(lastCommand.equals("*")) {
                result *= x;
            } else if(lastCommand.equals("/")) {
                result /= x;
            } else if(lastCommand.equals("=")) {
                result= x;
            }
            editText.setText(""+ result);
        }
    }
```

第 6 章

Android 弹出信息

在程序运行中，有些错误、关键信息需要提示给用户，有些程序判断步骤需要用户的参与，有些特殊信息需要用户的输入，等等，这都需要用到对话框和信息提示框。在 Android 中也是如此。这可以帮助我们实现更为人性化的软件。

Android 提供了 Toast 和 AlertDialog 两种方式来实现弹出信息。

6.1 Toast

Toast 是一种短暂的消息，它会自行显示和消失，不需要用户干预。而且，它不会从当前活动的 Activity 那里获取焦点，所以如果用户正忙于编写一部优秀的编程指南，那么他的输入不会被该消息打断。

由于 Toast 是短暂的，所以无法知道用户是否已注意到它。你不会得到任何确认，消息也不会出现太长时间以至于影响到用户。因此，Toast 通常用于建议性的消息，例如提示一个运行时间很长的后台任务已经完成，电池电量低(但不是太低)，等等。

构建 Toast 非常简单。Toast 类提供了一个静态 makeText()方法，它接受一个 String(或字符串资源 ID)并返回一个 Toast 实例。makeText()方法还需要 Activity(或其他 Context)以及一个持续时间。持续时间表示 LENGTH_SHORT 或 LENGTH_LONG 常量形式，以相对方式指示消息应该显示多久。

例如：

```
Toast.makeText(getApplicationContext(), "默认样式",
               Toast.LENGTH_SHORT).show();
```

Toast 有 5 种常用的显示形式，下面举例一一介绍。

(1) 默认效果，界面如图 6.1 所示。

(2) 自定义显示位置，效果如图 6.2 所示。

(3) 带图片效果，如图 6.3 所示。

(4) 完全自定义效果，如图 6.4 所示。

(5) 其他线程效果，如图 6.5 所示。

这 5 种效果对应的界面布局的 XML 文件如下：



图 6.1 默认效果



图 6.2 自定义显示位置效果

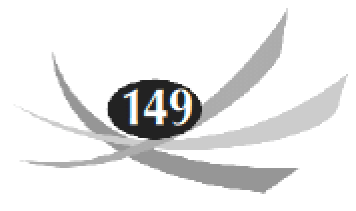




图 6.3　带图片效果



图 6.4　完全自定义效果



图 6.5 其他线程效果

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
  Android:orientation="vertical" Android:layout_width="fill_parent"
  Android:layout_height="fill_parent" Android:padding="5dip"
                    Android:gravity="center">
  <Button Android:layout_height="wrap_content"
    Android:layout_width="fill_parent" Android:id="@+id/toast"
    Android:text="默认效果展示 "></Button>
  <Button Android:layout_height="wrap_content"
    Android:layout_width="fill_parent" Android:text="自定义显示位置效果展示 "
    Android:id="@+id/toastCustomPosition"></Button>
  <Button Android:layout_height="wrap_content"
    Android:layout_width="fill_parent" Android:id="@+id/toastWithImage"
    Android:text="带图片效果展示"></Button>
  <Button Android:layout_height="wrap_content"
    Android:layout_width="fill_parent" Android:text="完全自定义效果展示"
    Android:id="@+id/toastCustom"></Button>
  <Button Android:layout_height="wrap_content"
    Android:layout_width="fill_parent" Android:text="其他线程效果展示"
    Android:id="@+id/toastFromOtherThread"></Button></LinearLayout>
```

完全自定义效果 Toast 通知窗口对应界面布局的 XML 文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:Android="http://schemas.Android.com/apk/res/Android"
```

```
Android:layout_height="wrap_content" Android:layout_width="wrap_content"
Android:background="#ffffffff" Android:orientation="vertical"
Android:id="@+id/custom" >
<TextView
  Android:layout_height="wrap_content"
  Android:layout_margin="1dip"
  Android:textColor="#ffffffff"
  Android:layout_width="fill_parent"
  Android:gravity="center"
  Android:background="#bb000000"
  Android:id="@+id/titlecus" />
<LinearLayout
  Android:layout_height="wrap_content"
  Android:orientation="vertical"
  Android:id="@+id/customcontent"
  Android:layout_marginLeft="1dip"
  Android:layout_marginRight="1dip"
  Android:layout_marginBottom="1dip"
  Android:layout_width="wrap_content"
  Android:padding="15dip"
  Android:background="#44000000" >
<ImageView
  Android:layout_height="wrap_content"
  Android:layout_gravity="center"
  Android:layout_width="wrap_content"
  Android:id="@+id/imagecus" />
  <TextView
  Android:layout_height="wrap_content"
  Android:paddingRight="10dip"
  Android:paddingLeft="10dip"
  Android:layout_width="wrap_content"
  Android:gravity="center"
  Android:textColor="#ff000000"
  Android:id="@+id/textcus" />
</LinearLayout>
</LinearLayout>
```

对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.os.Handler;
import Android.view.Gravity;
import Android.view.LayoutInflater;
```

```
import Android.view.View;
import Android.view.View.OnClickListener;
import Android.view.ViewGroup;
import Android.widget.Button;
import Android.widget.ImageView;
import Android.widget.LinearLayout;
import Android.widget.TextView;
import Android.widget.Toast;
public class ToastActivity extends Activity {
    Handler handler= new Handler();
    Button bt_toast= null;
    Button bt_toastposition= null;
    Button bt_toastimag= null;
    Button bt_toastcustom= null;
    Button bt_toastthread= null;
    Toast toast= null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.toast_dialog);
        bt_toast= (Button)super.findViewById(R.id.toast);
        bt_toastposition=
                    (Button)super.findViewById(R.id.toastCustomPosition);
        bt_toastimag= (Button)super.findViewById(R.id.toastWithImage);
        bt_toastcustom= (Button)super.findViewById(R.id.toastCustom);
        bt_toastthread=
                    (Button)super.findViewById(R.id.toastFromOtherThread);
        bt_toast.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(),
                        "默认 Toast 样式效果展示",Toast.LENGTH_SHORT).show();
            }
        });
        bt_toastposition.setOnClickListener(new OnClickListener(){
                @Override
                public void onClick(View v) {
                    toast= Toast.makeText(getApplicationContext(),
                            "自定义位置效果展示", Toast.LENGTH_LONG);
                    toast.setGravity(Gravity.CENTER, 0, 0);      //设定展示位置
                    toast.show();
                }
        });
```

```
bt_toastimag.setOnClickListener(new OnClickListener(){
        @Override
        public void onClick(View v) {
            toast= Toast.makeText(getApplicationContext(),
                    "带图片效果展示", Toast.LENGTH_LONG);
            toast.setGravity(Gravity.CENTER, 0, 0);
            LinearLayout toastView=
                    (LinearLayout) toast.getView();
            ImageView imageCodeProject=
                    new ImageView(getApplicationContext());                //新建图片控件
            imageCodeProject.setImageResource(R.drawable.icon);
                //设定展示位置
            toastView.addView(imageCodeProject, 0);                        //加载图片控件
            toast.show();
        }
});
bt_toastcustom.setOnClickListener(new OnClickListener(){
        @Override
        public void onClick(View v) {
            LayoutInflater inflater= getLayoutInflater();
            View layout= inflater.inflate(R.layout.toastcustom,
                        (ViewGroup) findViewById(R.id.custom));
            //设置新的 view的来源。LayoutInflater查找 Layout下的布局文件
            ImageView image=
                    (ImageView) layout.findViewById(R.id.imagecus);
            image.setImageResource(R.drawable.icon);                       //设置图片路径
            TextView title=
                        (TextView) layout.findViewById(R.id.titlecus);
            title.setText("Toast通知展示");
            TextView text=
                        (TextView) layout.findViewById(R.id.textcus);
            text.setText("完全自定义效果展示");
            toast= new Toast(getApplicationContext());
            toast.setGravity(Gravity.RIGHT | Gravity.TOP, 12, 40);
            toast.setDuration(Toast.LENGTH_LONG);                          //持续时间
            toast.setView(layout);
            toast.show();
        }
});
bt_toastthread.setOnClickListener(new OnClickListener(){
        @Override
        public void onClick(View v) {
            new Thread(new Runnable(){
```

```
                    public void run() {showToast();}
                }).start();
            }
        });
    }
    public void showToast() {
      handler.post(new Runnable() {
       @Override
       public void run() {
         Toast.makeText(getApplicationContext(), "Toast来自其他线程!",
            Toast.LENGTH_SHORT).show();
       }
      });
    }
}
```

6.2 AlertDialog

与 Toast 不一样，AlertDialog 将弹出并获取焦点，一直显示，直到被用户关闭。可以用这个提醒框来显示关键错误，或者让用户立即看到的消息，或者等待用户输入必须输入的信息等。

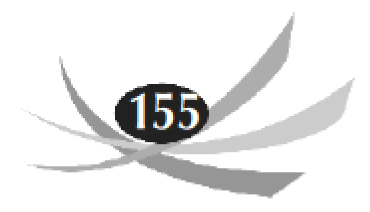
AlertDialog 的构造方法全部是 Protected 的，所以不能直接通过实例化一个 AlertDialog 来创建一个 AlertDialog，要用到 AlertDialog.Builder 中的 create()方法。

Builder 中常用方法如下所示。

- create ()：创建对话框。
- show()：显示对话框。
- setMessage()：将对话框的"主体"设置为一个简单的文本消息，来自所提供的 String 或所提供的字符串资源 ID。
- setTitle()：为对话框设置标题。
- setIcon()：为对话框设置标题栏图标。
- setView()：给对话框设置自定义样式。
- setItems()：设置对话框要显示的一个 list，一般用于显示几个命令时。
- setMultiChoiceItems ()：用来设置对话框显示一系列的复选框。
- setNeutralButton()：普通按钮。
- setPositiveButton()：给对话框添加 Yes 按钮。
- setNegativeButton()：给对话框添加 No 按钮。

最后 3 个方法可以设置按钮放在什么位置(分别放在左侧、中间或右侧)，这些按钮的显示名称应该是什么，以及单击按钮时应该调用什么逻辑(除了关闭对话框)。

下面分几种常见形式介绍对话框的实现方式。



(1) 当单击返回按钮时弹出一个提示,确保操作无误,采用常见的对话框样式,简单效果如图 6.6 所示。



图 6.6　简单效果

(2) 多个按钮,改变了对话框的图表,添加了 3 个按钮,如图 6.7 所示。



图 6.7　多按钮效果

(3) 带输入框的,信息内容是一个简单的 View 类型,如图 6.8 所示。

(4) 进度框,如图 6.9 所示。



图 6.8 输入框效果



图 6.9 进度框效果

(5) 列表效果,如图 6.10 所示。

用 setItems(CharSequence[] items, final OnClickListener listener)方法来实现类似 ListView 的 AlertDialog。

其中,第一个参数是要显示的数据的数组;第二个参数是单击某个 item 的触发事件。

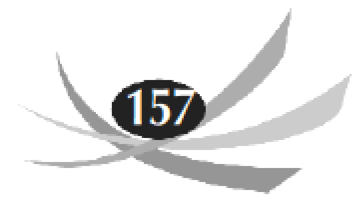




图 6.10　列表效果

(6) 单选列表效果,如图 6.11 所示。



图 6.11　单选列表效果

用 setSingleChoiceItems(CharSequence[] items, int checkedItem, final OnClickListener listener)方法来实现类似 RadioButton 的 AlertDialog。

其中,第一个参数是要显示的数据的数组;第二个参数是初始值(初始被选中的item);第三个参数是单击某个 item 的触发事件。

(7) 复选列表效果,如图 6.12 所示。



图 6.12 复选列表效果

用 setMultiChoiceItems(CharSequence[] items, boolean[] checkedItems, final OnMultiChoiceClickListener listener)方法来实现类似 CheckBox 的 AlertDialog。

其中,第一个参数是要显示的数据的数组;第二个参数是选中状态的数组;第三个参数是单击某个 item 的触发事件。

(8) 自定义 View 效果,如图 6.13 所示。

以上各种效果对应的布局 Layout 文件就是一个多按钮的界面。大家可以自己去实现。

自定义 View 中的 View 的布局 Layout 文件,就是一个登录信息维护,代码如下:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:layout_width="fill_parent"
    Android:layout_height="wrap_content"
    Android:orientation="vertical">
    <TextView Android:id="@+id/username_view"
        Android:layout_height="wrap_content"
```



图 6.13 自定义效果

```
    Android:layout_width="wrap_content"
     Android:layout_marginLeft="20dip"
    Android:layout_marginRight="20dip" Android:text="请输入用户名:"
    Android:textAppearance="?Android:attr/textAppearanceMedium" />
<EditText Android:id="@+id/username_edit"
    Android:layout_height="wrap_content"
    Android:layout_width="fill_parent"
     Android:layout_marginLeft="20dip"
    Android:layout_marginRight="20dip" Android:capitalize="none"
    Android:textAppearance="?Android:attr/textAppearanceMedium" />
<TextView Android:id="@+id/password_view"
    Android:layout_height="wrap_content"
    Android:layout_width="wrap_content"
     Android:layout_marginLeft="20dip"
    Android:layout_marginRight="20dip" Android:text="请输入密码:"
    Android:textAppearance="?Android:attr/textAppearanceMedium" />
<EditText Android:id="@+id/password_edit"
    Android:layout_height="wrap_content"
    Android:layout_width="fill_parent"
        Android:layout_marginLeft="20dip"
    Android:layout_marginRight="20dip" Android:capitalize="none"
    Android:password="true"
    Android:textAppearance="?Android:attr/textAppearanceMedium" />
```

```
</LinearLayout>
```

对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.app.AlertDialog;
import Android.app.Dialog;
import Android.app.ProgressDialog;
import Android.content.Context;
import Android.content.DialogInterface;
import Android.os.Bundle;
import Android.view.LayoutInflater;
import Android.view.View;
import Android.view.View.OnClickListener;
import Android.widget.Button;
import Android.widget.EditText;
import Android.widget.Toast;
public class DialogActivity extends Activity {
    private static final int DIALOG1= 1;
    private static final int DIALOG2= 2;
    private static final int DIALOG3= 3;
    private static final int DIALOG4= 4;
    private static final int DIALOG5= 5;
    private static final int DIALOG6= 6;
    private static final int DIALOG7= 7;
    private static final int DIALOG8= 8;
    @Override
    protected Dialog onCreateDialog(int id) {
        switch(id) {
        case DIALOG1:
            return buildDialog1(DialogActivity.this);
        case DIALOG2:
            return buildDialog2(DialogActivity.this);
        case DIALOG3:
            return buildDialog3(DialogActivity.this);
        case DIALOG4:
            return buildDialog4(DialogActivity.this);
        case DIALOG5:
            return buildDialog5(DialogActivity.this);
        case DIALOG6:
            return buildDialog6(DialogActivity.this);
        case DIALOG7:
            return buildDialog7(DialogActivity.this);
        case DIALOG8:
            return buildDialog8(DialogActivity.this);
```

```
        }
        return null;
    }
    protected void onPrepareDialog(int id, Dialog dialog){
        if(id==DIALOG1){
            setTitle("测试");
        }
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.alert_dialog);
        Button button1= (Button) findViewById(R.id.button1);
        button1.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                showDialog(DIALOG1);
            }
        });
        Button buttons2= (Button) findViewById(R.id.buttons2);
        buttons2.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                showDialog(DIALOG2);
            }
        });
        Button button3= (Button) findViewById(R.id.button3);
        button3.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                showDialog(DIALOG3);
            }
        });
        Button button4= (Button) findViewById(R.id.button4);
        button4.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                showDialog(DIALOG4);
            }
        });
        Button button5= (Button) findViewById(R.id.button5);
        button5.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                showDialog(DIALOG5);
            }
        });
        Button button6= (Button) findViewById(R.id.button6);
        button6.setOnClickListener(new OnClickListener() {
```

```
                public void onClick(View v) {
                    showDialog(DIALOG6);
                }
            });
            Button button7= (Button) findViewById(R.id.button7);
            button7.setOnClickListener(new OnClickListener() {
                public void onClick(View v) {
                    showDialog(DIALOG7);
                }
            });
            Button button8= (Button) findViewById(R.id.button8);
            button8.setOnClickListener(new OnClickListener() {
                public void onClick(View v) {
                    showDialog(DIALOG8);
                }
            });
        }
        private Dialog buildDialog1(Context context) {
            AlertDialog.Builder builder= new AlertDialog.Builder(context);
            builder.setIcon(R.drawable.alert_dialog_icon);
            builder.setTitle("标题自定义");
            builder.setPositiveButton(R.string.alert_dialog_ok,
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog,
                                                                int whichButton) {
                            setTitle("单击了对话框上的--确定--按钮");
                        }
                    });
            builder.setNegativeButton(R.string.alert_dialog_cancel,
                        new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog,
                            int whichButton) {
                            setTitle("单击了对话框上的--取消--按钮");
                        }
                    });
            return builder.create();

        }
        private Dialog buildDialog2(Context context) {
            AlertDialog.Builder builder= new AlertDialog.Builder(context);
            builder.setIcon(R.drawable.alert_dialog_icon);
            builder.setTitle("标题自定义-这是多按钮展示");
            builder.setMessage("这是多按钮展示");
            builder.setPositiveButton(R.string.alert_dialog_ok,
```

```
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                                int whichButton) {
                        setTitle("单击了对话框上的--确定--按钮");
                    }
                });
        builder.setNeutralButton(R.string.alert_dialog_something,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                                    int whichButton) {
                        setTitle("单击了对话框上的--进入详细--按钮");
                    }
                });
        builder.setNegativeButton(R.string.alert_dialog_cancel,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                                int whichButton) {
                        setTitle("单击了对话框上的--取消--按钮");
                    }
                });
        return builder.create();
}
private Dialog buildDialog3(Context context) {
        return new AlertDialog.Builder(this).setTitle("请输入").setIcon(
                Android.R.drawable.ic_dialog_info).setView(
                new EditText(this)).setPositiveButton("确定", null)
                .setNegativeButton("取消", null).show();
}
private Dialog buildDialog4(Context context) {
        ProgressDialog dialog= new ProgressDialog(context);
        dialog.setTitle("正在下载歌曲");
        dialog.setMessage("请稍候……");
        return  dialog;
}
private Dialog buildDialog5(Context context) {
         final String[] arrayFruit= new String[] {"苹果", "橘子","草莓", "香蕉"};
          Dialog alertDialog= new AlertDialog.Builder(this).
            setTitle("你喜欢吃哪种水果?").
            setIcon(R.drawable.alert_dialog_icon)
            .setItems(arrayFruit, new DialogInterface.OnClickListener(){
             @Override
             public void onClick(DialogInterface dialog, int which) {
               Toast.makeText(DialogActivity.this, arrayFruit[which],
                                    Toast.LENGTH_SHORT).show();
```

```
            }}).
            setNegativeButton("取消",new DialogInterface.OnClickListener(){
            @Override
            public void onClick(DialogInterface dialog, int which) {
              //TODO Auto-generated method stub
            }}).create();
        return  alertDialog;
    }
    private int selectedFruitIndex=0;
    private Dialog buildDialog6(Context context) {
        final String[] arrayFruit=new String[] {"苹果", "雪梨", "木瓜",
                                              "香蕉", "橘子"};
          Dialog alertDialog=new AlertDialog.Builder(this).
            setTitle("你喜欢吃哪种水果?").
            setIcon(R.drawable.alert_dialog_icon)
            .setSingleChoiceItems(arrayFruit, 0,
                                              new DialogInterface.OnClickListener(){
            @Override
            public void onClick(DialogInterface dialog, int which) {
              selectedFruitIndex=which;
            }}).
            setPositiveButton("确认",new DialogInterface.OnClickListener(){
            @Override
            public void onClick(DialogInterface dialog, int which) {
              Toast.makeText(DialogActivity.this,
                         arrayFruit[selectedFruitIndex],
                           Toast.LENGTH_SHORT).show();
            }}).
            setNegativeButton("取消",new DialogInterface.OnClickListener(){
            @Override
            public void onClick(DialogInterface dialog, int which) {
            }}).create();
        return  alertDialog;
    }
    private Dialog buildDialog7(Context context) {
        final String[] arrayFruit=new String[] {"苹果", "雪梨", "木瓜",
                                              "香蕉", "橘子"};
        final boolean[] arrayFruitSelected=new boolean[] {true, true, false,
                                                  false, false};
        Dialog alertDialog=new AlertDialog.Builder(this).
                setTitle("你喜欢吃哪种水果?").
                setIcon(R.drawable.alert_dialog_icon)
                .setMultiChoiceItems(arrayFruit, arrayFruitSelected,
                          new DialogInterface.OnMultiChoiceClickListener() {
```

```
                    @Override
                    public void onClick(DialogInterface dialog, int which,
                                                boolean isChecked) {
                        arrayFruitSelected[which]= isChecked;
                    } }).
                setPositiveButton("确认",
                              new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        StringBuilder stringBuilder= new StringBuilder();
                        for(int i= 0; i < arrayFruitSelected.length; i++) {
                            if(arrayFruitSelected[i]= = true)
                            {
                                stringBuilder.append(arrayFruit[i]+ "、");
                            }
                        }
                        Toast.makeText(DialogActivity.this,
                                  stringBuilder.toString(),
                                  Toast.LENGTH_SHORT).show();
                    } }).
                setNegativeButton("取消",
                                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                    } }).create();
        return   alertDialog;
    }
    private Dialog buildDialog8(Context context) {
        LayoutInflater inflater= LayoutInflater.from(this);
        final View textEntryView= inflater.inflate(
                R.layout.alert_dialog_text_entry, null);
        AlertDialog.Builder builder= new AlertDialog.Builder(context);
        builder.setIcon(R.drawable.alert_dialog_icon);
        builder.setTitle(R.string.alert_dialog_text_entry);
        builder.setView(textEntryView);
        builder.setPositiveButton(R.string.alert_dialog_ok,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                              int whichButton) {
                        setTitle("单击了对话框上的--确定--按钮");
                    }
                });
        builder.setNegativeButton(R.string.alert_dialog_cancel,
                new DialogInterface.OnClickListener() {
```

```
                        public void onClick(DialogInterface dialog,
                                int whichButton) {
                            setTitle("单击了对话框上的--取消--按钮");
                        }
                    });
            return builder.create();
        }
    }
```

第 7 章 Android 资源

在传统的 Java 程序中，一般都把在程序中用到的字符串、常量直接写在代码中或维护在数据库中。写在代码里尽管会给当时的编程带来便利，但是会大大增强后期的维护成本。即使维护在数据库中也会增加编码的工作量。

Android 对这种情况做了改进，可以将程序中用到的各种资源，例如字符串、颜色、数组、菜单、图片、声音、视频等都放到 res 目录中定义，不需要数据库，实现了资源的灵活维护。这样做既减少了代码量，也为程序的后期维护带来便利，可以在不重新修改源代码的情况下实现资源的更新。

Android 资源分为系统资源和用户资源。

Android 系统提供了大量的系统资源，这些资源都放在 SDK 中的/platforms/Android-版本/data/res 目录中。在 Java 代码中可以使用“Android. R. 资源文件种类. 资源 ID”访问。在 XML 文件中使用“@Android:资源种类/资源 ID”访问。

资源的种类如表 7-1 所示。

表 7-1　res 目录结构

目录	资源类型	描　　述
res/anim	xml	动画信息、帧动画或者补间动画
res/layout	xml	保存布局信息
res/values	xml	字符串、颜色、尺寸、类型、主题等。文件可以任意命名，采用 key-value 的形式，建议不同的文件保存不同类型的值
res/menu	xml	保存菜单资源，一个资源文件表示一个菜单
res/xml	xml	用来保存任意 xml 文件，可以通过代码 Resources. getXML()来读取
res/raw	任意类型	目录中的资源不会被编译，可以调用 Resource. openRawResource(int id)获得资源的二进制输入流
assets	任意类型	与 raw 一样，不会被编译，不同的是该目录中的资源文件不会生成资源 ID
res/drawable	图形	保存多种格式的图像

7.1　res/values

这是资源中比较重要的一个文件夹，包含多种类型的文件。文件可以被命名为任何名字，文件夹中有以下典型的文件(一般约定文件以定义的元素类型为文件名)。

7.1.1 strings.xml

strings.xml 用于定义字符串值。

字符串资源文件位于 res/values 目录下，该资源文件的根节点是<resources>，每个<string>元素代表的是一个字符串常量，其中，name 为常量名，标签中间的内容为字符串的值。每个<string-array>元素代表的是一个字符串数组常量。其中，name 表示常量名，<item>标签代表的是数组的一个数组项。

在 XML 文件中访问字符串的<string>内容使用如下语法：

```
@[package:]string/name
```

在 XML 文件中访问字符串的<string-array>内容使用如下语法：

```
@[package:]array/name
```

在 Java 文件中访问资源文件中的<string>内容使用如下语法：

```
getResource().getString(id)
getResource().getText(id)
```

在 Java 文件中访问资源文件中的<string-array>内容使用如下语法：

```
Resources res=getResources();
String[] arrays=res.getStringArray(R.array.supplier_style);
```

如果想在字符串中使用引号(单引号或者双引号)，就使用转义符号"\"，否则引号会被忽略。

使用占位符来实现动态的字符串资源信息：

```
<string name="dyna">大家好,我是%1$s,今年%2$d岁</string>
```

其中，%1 和%2 代表索引的位置，必须从 1 开始；$s 和 $d 分别代表字符串和十进制数字类型的变量。

代码中可以以如下方式获取该字符信息并指定变量的值：

```
dynatext=(TextView)findViewById(R.id.dynaText);
dynatext.setText(getResources().getString(R.string.dyna, "张三疯",35));
```

getString 方法的第二个参数是一个可变参数，也就意味着可以传递任意多个参数值，使用 Resources.getString()或 Resources.getText()取得资源或者更实用。getText()能取得在用户界面上显示的文本框中的文本。

下面举例说明用法(实现界面在此不再提供)。

字符串值定义 XML 文件 strings.xml 如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Test strings Resource</string>
```

```
    <string name="test_str1">从代码中引用</string>
    <string name="test_str2">从资源文件中引用</string>
    <string name="value1">你好\'这是一个单引号效果</string>
    <string name="value2">"你好'这是一个双引号效果"</string>
    <string name="value3">\"你好\'这是一个带转义字符的双引号效果\"</string>
    <string-array name="supplier_style">
        <item>食品</item>
        <item>日化</item>
        <item>五金</item>
        <item>饮料</item>
        <item>服装</item>
        <item>医药</item>
        <item>其他</item>
    </string-array>
</resources>
```

使用 strings.xml 资源的布局文件 main.xml 代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:layout_width="fill_parent"
    Android:layout_height="fill_parent"
    Android:orientation="vertical" >
    <TextView
        Android:id="@+id/MyTextView01"
        Android:layout_width="fill_parent"
        Android:layout_height="wrap_content"
      Android:text="@string/test_str2"/>              //从资源中引用
    <TextView
        Android:id="@+id/MyTextView02"
        Android:layout_width="fill_parent"
        Android:layout_height="wrap_content"
        Android:text=""/>
    <TextView
    Android:layout_width="fill_parent"
    Android:textSize="30sp"
    Android:layout_height="wrap_content"
    Android:text="@string/value1" />
    <Spinner   Android:id="@+id/supplier_add_style_spinner"
               Android:layout_width="fill_parent"
               Android:layout_height="wrap_content"/>
</LinearLayout>
```

实现功能的 Activity 文件代码如下：

```
import Android.app.Activity;
```

```
import Android.os.Bundle;
import Android.widget.TextView;
public class StringActivity extends Activity {
    private TextView myTextView;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        myTextView= (TextView)findViewById(R.id.MyTextView02);
        String str= getString(R.string.test_str1).toString();
        myTextView.setText(str);
        ArrayAdapter<CharSequence> arrayAdapter=
            ArrayAdapter.createFromResource(this, R.array.supplier_style,
          Android.R.layout.simple_spinner_item);
    }
}
```

7.1.2 arrays.xml

1. Integer Array

用 XML 格式定义的整数数组。

资源引用：

在 Java 代码中引用格式：R.array.string_array_name。

在 Java 代码中取出 integer array 的应用程序代码如下：

```
Resources res= getResources();
int[] bits= res.getIntArray(R.array.bits);
```

在 XML 代码中引用格式：@[package:]array/integer_array_name。

数组定义 XML 文件代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<integer-array name="bits">
<item>4</item>
<item>8</item>
<item>16</item>
<item>32</item>
</integer-array>
</resources>
```

2. Typed Array

用 XML 格式定义的 TypedArray，用于创建其他资源的数组，例如 drawable。注意：数组元素不必是同一类型的，可以创建多种资源组成的数组。但必须小心处理数组内不

同的数据类型,利用 TypedArray 的 get…()属性正确地读取每个数据项。

资源引用:

在 Java 代码中引用格式:R.array.array_name

在 Java 代码中取出每个数组并读取第一个数组元素:

```
Resources res= getResources();
TypedArray icons= res.obtainTypedArray(R.array.icons);
Drawable drawable= icons.getDrawable(0);
TypedArray colors= res.obtainTypedArray(R.array.icons);
int color= colors.getColor(0,0);
```

在 XML 代码中引用格式:@[package:]array.array_name

数组定义的 XML 文件代码如下:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<array name="icons">
<item>@drawable/home</item>
<item>@drawable/settings</item>
<item>@drawable/logout</item>
</array>
<array name="colors">
<item>#FFFF0000</item>
<item>#FF00FF00</item>
<item>#FF0000FF</item>
</array>
</resources>
```

7.1.3 Bools.xml

Bools.xml 是用 XML 格式定义的布尔值。

注意:bool 是简单类型资源,是用名称(name)属性(而非 XML 文件名)直接引用的。因此,在一个 XML 文件中,可以把 bool 资源和其他简单类型资源一起放入一个 <resources>元素下。

定义资源的 XML 文件代码如下:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<bool name="screen_small">true</bool>
<bool name="adjust_view_bounds">true</bool>
</resources>
```

以下应用程序代码取出 bool 值:

```
Resources res= getResources();
```

```
boolean screenIsSmall= res.getBoolean(R.bool.screen_small);
```

以下布局 XML 文件将 bool 资源用于属性：

```
<ImageView
Android:layout_height="fill_parent"
Android:layout_width="fill_parent"
Android:src="@drawable/logo"
Android:adjustViewBounds="@bool/adjust_view_bounds" />
```

7.1.4 colors.xml

colors.xml 定义颜色和颜色字串数值。

Android 允许将颜色值作为资源保存在资源文件中，保存在资源文件中的颜色值以#开始，系统支持#RGB、#ARGB、#RRGGBB、#AARRGGBB 这 4 种形式的表示方法。

A 表示透明度，R、G、B 分别代表红绿蓝三原色。A 的值可以省略，如果省略，那么颜色则是完全不透明的。

如果采用前面两种表示法，A、R、G、B 的取值为 0～15；如果采用后两种表示法，则取值范围都是 0～255，A 取 0 表示完全透明，255 为完全不透明，R、G、B 取值越大，则代表颜色越深。

在 XML 中通过@color/****来访问颜色资源的值。

代码中则用 getResources().getColor(R.color.***)或者 getResources().getColor(id)。

可以使用 Resources.getDrawable()以及 Resources.getColor()、respectively 取得这些资源。

下面举例说明用法。

颜色值定义的 XML 文件 colors.xml 如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<color name="red_text">#f00</color>
<color name="blue_bag">#0000ff</color>
</resources>
```

展示使用资源的布局文件代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:layout_width="fill_parent"
    Android:layout_height="fill_parent"
    Android:orientation="vertical" >
    <TextView
        Android:text="测试颜色资源文件"
```

```
        Android:textColor= "@ color/red_text"    //引用颜色资源,在资源文件下引用的
            Android:layout_width= "fill_parent"
            Android:layout_height= "wrap_content"/>
</LinearLayout>
```

实现功能的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
public class ColorActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    //map 键--值    savedInstanceState:保存状态
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        getWindow().setBackgroundDrawableResource(R.color.blue_bag);
                        //改变背景颜色
    }
}
```

7.1.5　dimens.xml

dimens.xml 定义尺寸数据。

尺寸资源是用来定义大小的,由一系列的浮点数组成,尺寸资源要在 res/values 目录下的资源文件中借助<dimen>标签定义。

资源引用：

在 Java 代码中引用格式：R.dimen.name。

在 Java 代码中取出尺寸的应用程序代码如下：

```
float btn_h= getResources().getDimension(R.dimen.name);
```

在 XML 代码中引用格式：@dimen/ name。

Android 支持的度量单位有以下几种。

- dp：分辨率无关的像素(Pixel)单位,是基于屏幕的物理(像素)分辨率的抽象单位。此单位基于一个 160dpi(每英寸点数)的屏幕,所以 160dp 常常是 1 英寸且与屏幕像素分辨率无关。dp 和像素的比率会随着屏幕密度而变化,但不一定成正比。建议用于在 layout 中指定 View 尺寸,这样 UI 在不同屏幕上能自动缩放而显示出相同的大小(dpi 和 dp 同义,编译器都可接受,虽然 dp 更近似于 sp)。
- sp：缩放无关的像素单位,类似于 dp,但还会根据用户的字体大小设置进行缩放。建议用于指定字体大小,这样根据屏幕分辨率和用户设置都能自动调整。
- pt：点,基于屏幕实际尺寸,对应 1/72 英寸。
- px：像素,与屏幕实际像素一致。这是个不建议使用的单位,因为在不同设备上

的实际表现差异很大，每种设备每英寸的像素数可能不同，屏幕上的总像素数也可能更多或更少。

- mm：毫米，基于屏幕物理尺寸。
- in：英寸，基于屏幕物理尺寸。

定义资源 XML 的文件代码如下：

```
<dimen name="dimen1">40dp</dimen>
<dimen name="dimen2">20px</dimen>
```

下面举例说明用法。

颜色值定义的 XML 文件 dimens.xml 如下：

```
<?xml version="1.0" encoding="utf-8"?>
 <resources>
 <dimen name="text_width">150px</dimen>
 <dimen name="text_heigh">100px</dimen>
 <dimen name="btn_width">30mm</dimen>
 <dimen name="btn_heigh">10mm</dimen>
</resources>
```

展示使用资源的布局文件代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:layout_width="fill_parent"
    Android:layout_height="fill_parent"
    Android:orientation="vertical" >
    <TextView
        Android:id="@+id/myDimenTextView01"
        Android:layout_width="fill_parent"
        Android:layout_height="wrap_content"
        Android:width="@dimen/text_width"
        Android:height="@dimen/text_heigh"
        Android:text="@string/hello"
        Android:background="#f00"
        />
    <Button
         Android:id="@+id/Button01"
         Android:layout_width="wrap_content"
         Android:layout_height="wrap_content"
         Android:text="按钮"/>
</LinearLayout>
```

实现功能的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
```

```
import Android.content.res.Resources;
import Android.os.Bundle;
import Android.widget.Button;
public class TestdimensActivity extends Activity {
    //第一步:定义按钮名称
    private Button MyButton;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //第二步:对应按钮
        MyButton= (Button)findViewById(R.id.Button01);
        Resources r= getResources();          //第三步:获得尺寸资源
        float btn_h= r.getDimension(R.dimen.btn_heigh);
        float btn_w= r.getDimension(R.dimen.btn_width);
        //第四步:利用尺寸资源
        MyButton.setWidth((int)btn_w);
        MyButton.setHeight((int)btn_h);
    }
}
```

7.1.6 ids.xml

ids.xml定义资源ID数据。

ids.xml是用XML格式定义的资源唯一ID。请记住ID资源不代表一个实际的资源项,而只是一个可与其他资源绑定的唯一ID,或是一个用于应用程序代码中的唯一整数。

资源引用如下:

在Java代码中引用格式:R.id.name。

在XML代码中引用格式:@[package:]id/name。

定义资源XML的文件代码如下:

```
<?xml version= "1.0" encoding= "utf-8"?>
<resources>
<item type= "id" name= "button_ok" />
<item type= "id" name= "dialog_exit" />
</resources>
```

以下布局段将“button_ok”用于Button控件ID:

```
<Button Android:id= "@ id/button_ok"
style= "@ style/button_style" />
```

注意:Android:id的值的ID引用中不含加号“+”了,因为这个ID已经在上面的ids.xml中定义过了。(如果XML资源中用加号指定一个ID——类似格式Android:

id="@+id/name"——那就意味着"name"命名的 ID 还不存在并需要创建它。)

以下代码段示例用 dialog_exit 作为对话框的唯一标识:

```
showDialog(R.id.dialog_exit);
```

7.1.7 styles.xml

styles. xml 定义类型对象。

样式资源可以将需要设置相同属性的属性值提取出来放到单独的文件中,然后在需要用到的地方引用该样式。这样可以保证界面风格的统一,同时也为修改样式带来了方便。这一点和以前的 CSS 技术相似。

样式资源也需要在 res/values 目录的资源文件中定义。每一个<style>标签表示一个样式,name 属性表示样式名,每个样式的属性使用 item 表示。样式可以继承。通过<style>的 parent 属性指定父样式资源 id。

定义样式资源的 XML 文件如下:

```
<style name="style1">
  <item name="Android:textColor">#ff0099</item>
  <item name="Android:textSize">25sp</item>
</style>
<style name="style2" parent="@style/style1">
    <item name="Android:gravity">right</item>
    <item name="Android:textSize">35sp</item>
</style>
```

在 XML 代码中引用样式的格式如下:

```
<TextView
        Android:layout_width="fill_parent"
        Android:layout_height="wrap_content"
        style="@style/style1"
        Android:text="这个是样式 1"/>
<TextView
        Android:layout_width="fill_parent"
        Android:layout_height="wrap_content"
        style="@style/style2"
        Android:text="这个是样式 2"/>
```

与样式资源类似,主题资源的 XML 文件也放在 res/values 下面,使用<resource>作为根元素,使用<style>来定义主题,与样式资源的区别在于,主题不能作用于单个的 View,主题可以对整个应用的 Activity 起作用,或对指定的 Activity 起作用。

只能在<application>、<activity>标签利用 Android:theme 属性设定主题资源。

定义主题资源的 XML 文件如下:

```
<style name="theme1">
```

```
< item name= "Android:background"> @ drawable/wallpaper< /item>
< item name= "Android:textColor"> # 3399ff< /item>
< item name= "Android:textSize"> 20sp< /item>
   < /style>
```

在 XML 代码中引用主题的格式如下：

```
< activity Android:theme= "@ style/theme1"
          Android:name= ".ThemeActivity1"> < /activity>
```

7.2　res/drawable

drawable 文件夹存放图片文件。

下面举例说明用法。

展示使用资源的布局 XML 文件代码如下：

```
< ?xml version= "1.0" encoding= "utf- 8"?>
< LinearLayout xmlns:Android= "http://schemas.Android.com/apk/res/Android"
     Android:layout_width= "fill_parent"
     Android:layout_height= "fill_parent"
     Android:orientation= "vertical" >
     < TextView
          Android:id= "@ + id/bitmapTextView"
          Android:layout_width= "fill_parent"
          Android:layout_height= "wrap_content"
          Android:text= "@ string/hello" />
     < ImageView
          Android:id= "@ + id/bitmapImageView01"
          Android:layout_width= "wrap_content"
          Android:layout_height= "wrap_content"
          Android:src= "@ drawable/a"
       />
     < ImageView Android:id= "@ + id/bitmapImageView02"
          Android:layout_width= "wrap_content"
          Android:layout_height= "wrap_content"/>
< /LinearLayout>
```

实现功能的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.content.res.Resources;
import Android.graphics.drawable.Drawable;
import Android.os.Bundle;
import Android.widget.ImageView;
public class TestdrawableActivity extends Activity {
```

```
    //第一步:定义控件
private ImageView myImageView;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    //第二步:连接控件
        myImageView= (ImageView)findViewById(R.id.bitmapImageView02);
    //第三步:获取图片资源
        Resources r= getResources();
        Drawable d= r.getDrawable(R.drawable.b);
    //第四步:设置图片资源
        myImageView.setImageDrawable(d);
    }
}
```

7.3 res/xml

文件夹中存放 xml 文件。

在 Java 代码中使用 getResource().gerXML()来获取 XML 文件。

下面举例说明用法。

定义一个 xml 文件,存放在 res/xml 文件夹中。内容如下:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<customer name="tom" age="20" gender="male" email="tom@yahoo.com"/>
<customer name="kite" age="21" gender="male" email="kite@yahoo.com"/>
</resources>
```

使用这个 xml 资源的布局文件(存放在 layout 文件夹中)内容如下:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:layout_width="fill_parent"
    Android:layout_height="fill_parent"
    Android:orientation="vertical" >
    <TextView
        Android:id="@+id/xmlContentTextView"
        Android:layout_width="fill_parent"
        Android:layout_height="wrap_content"
        Android:text=""
        Android:background="#f00"/>
    <Button
```

```
        Android:id="@+id/xmlTestButton"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:text="获取 xml 内容"/>
</LinearLayout>
```

实现引用 XML 资源功能的 Activity 的 Java 代码如下：

```
import Java.io.IOException;
import org.xmlpull.v1.XmlPullParserException;
import Android.app.Activity;
import Android.content.res.Resources;
import Android.content.res.XmlResourceParser;
import Android.os.Bundle;
import Android.view.View;
import Android.view.View.OnClickListener;
import Android.widget.Button;
import Android.widget.TextView;
public class XmlActivity extends Activity {
    private TextView myTextView;
    private Button myButton;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        myTextView=(TextView) findViewById(R.id.xmlContentTextView);
        myButton=(Button) findViewById(R.id.xmlTestButton);
        myButton.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                //TODO Auto-generated method stub
                int counter=0;
                StringBuilder sb=new StringBuilder("");
                Resources r=getResources();
                XmlResourceParser xrp=r.getXml(R.xml.test);
                try {
                    while(xrp.getEventType()!=
                                XmlResourceParser.END_DOCUMENT)
                    {
                        if(xrp.getEventType()==
                                XmlResourceParser.START_TAG)
                        {
                            String name=xrp.getName();
                            if(name.equals("customer"))
                            {
```

```
                        counter++;
                        sb.append("第"+counter+"条客户信息"+"\n");
                        sb.append(xrp.getAttributeValue(0)+"\n");
                        sb.append(xrp.getAttributeValue(1)+"\n");
                        sb.append(xrp.getAttributeValue(2)+"\n");
                        sb.append(xrp.getAttributeValue(3)+"\n");
                    }
                }else if(xrp.getEventType()==
                            XmlResourceParser.END_TAG)
                {}
                else if(xrp.getEventType()==
                            XmlResourceParser.TEXT)
                {}
                try {
                    xrp.next();
                } catch(IOException e) {
                    //TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
            myTextView.setText(sb.toString());
        } catch(XmlPullParserException e) {
            //TODO Auto-generated catch block
            e.printStackTrace();
        }
      }
    });
  }
}
```

7.4 res/menu

文件夹中存放自定义的菜单资源。

菜单的具体功能介绍见第 8 章。本节提供一个菜单的实现以展示使用资源设置菜单的功能。

定义的存放在菜单的资源 XML 文件 menus.xml 如下：

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:Android="http://schemas.Android.com/apk/res/Android">
<item Android:title="文件/File"
      Android:icon="@drawable/file">
   <menu>
      <group Android:id="@+id/noncheckabel_group"
```

```
                    Android:checkableBehavior= "none">
                <item Android:id= "@+id/newFile"
                    Android:title= "新建"
                    Android:alphabeticShortcut= "n"/>
                <item Android:id= "@+id/openFile"
                    Android:title= "打开"
                    Android:alphabeticShortcut= "o"/>
                <item Android:id= "@+id/saveFile"
                    Android:title= "保存"
                     Android:alphabeticShortcut= "s"
                    />
            </group>
        </menu>
</item>
    <item  Android:title= "编辑"
            Android:icon= "@drawable/edit">
        <menu>
            <group Android:id= "@+id/edit_group"
                Android:checkableBehavior= "single">
                <item Android:id= "@+id/cut"
                    Android:title= "剪切"/>
                <item Android:id= "@+id/copy"
                    Android:title= "复制"/>
                <item Android:id= "@+id/paste"
                    Android:title= "粘贴"/>
            </group>
        </menu>
    </item>
<item Android:title= "帮助"
    Android:icon= "@drawable/help">
    <menu>
        <group Android:id= "@+id/help_group"
            >
            <item Android:id= "@+id/about"
                Android:title= "关于"/>
            <item Android:id= "@+id/exit"
                Android:title= "退出"/>
        </group>
    </menu>
</item>
</menu>
```

展示使用菜单资源的布局 XML 文件介绍如下：

```
<?xml version= "1.0" encoding= "utf-8"?>
```

```
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:layout_width="fill_parent"
    Android:layout_height="fill_parent"
    Android:orientation="vertical" >
    <TextView
        Android:id="@+id/menuTextView01"
        Android:layout_width="fill_parent"
        Android:layout_height="wrap_content"
        Android:text="@string/hello" />
</LinearLayout>
```

实现菜单功能的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.app.AlertDialog;
import Android.content.DialogInterface;
import Android.os.Bundle;
import Android.view.Menu;
import Android.view.MenuInflater;
import Android.view.MenuItem;
public class TestmenuActivity extends Activity {
//定义了一个变量
    private MenuInflater mi;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                    //实例化
      mi=new MenuInflater(this);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //TODO Auto-generated method stub                //对应 xml 文件
        mi.inflate(R.menu.file_menu, menu);
        return true;
}
//实现部分菜单功能的响应
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        //TODO Auto-generated method stub
        switch(item.getItemId()){
        case R.id.about:
            //应该有一个关于对话框
            aboutAlert("本案例演示的是如何使用 XML 菜单资源定义菜单");
            break;
```

```
            case R.id.exit:
                //应该有一个文本框,提示"是否要退出"
                exitAlert("真的要退出!");
                break;
            }
            return super.onOptionsItemSelected(item);
        }
        //显示"退出"对话框
        private void exitAlert(String msg){
            AlertDialog.Builder builder= new AlertDialog.Builder(this);
            builder.setMessage(msg).setCancelable(false).setPositiveButton(
                    "确定", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    //TODO Auto- generated method stub
                    finish();
                }
            }).setNegativeButton("取消",
                                        new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    //TODO Auto- generated method stub
                    return;
                }
            });
            AlertDialog alter= builder.create();
            alter.show();
        }
        //显示"关于"对话框
        private void  aboutAlert(String msg){
            AlertDialog.Builder builder= new AlertDialog.Builder(this);
            builder.setMessage(msg).setCancelable(false).setPositiveButton(
                        "确定", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
                    //TODO Auto- generated method stub
                }
            });
            AlertDialog alter= builder.create();
            alter.show();
        }
    }
```

7.5 res/raw

raw 目录用于存放应用程序使用的各种通用文件,例如音频文件等。安装到手机时,这些文件直接复制到手机中。raw 目录不可以有目录结构,下面不能再建目录。

这些资源会在 R 文件中生成对应的 id，程序中需要通过 id 来使用这些资源：

```
getResources().openRawResource(id)
```

通过返回的 InputStream 对象对文件数据进行读取数据：

```
InputStream is=getResources().openRawResource(R.id.filename)
```

读取文件的 Activity 的 Java 代码如下：

```
void readRawFile()
{
    String content;
    Resources resources=this.getResources();
    InputStream is=null;
    try{
        is=resources.openRawResource(R.raw.hubin);
        byte buffer[]=new byte[is.available()];
        is.read(buffer);
        content=new String(buffer);
        Log.i(tag, "read:"+content);
    }
    catch(IOException e)
    {
        Log.e(tag, "write file",e);
    }
    finally
    {
        if(is!=null)
        {
            try{
            is.close();
            }catch(IOException e)
            {
                Log.e(tag, "close file",e);
            }
        }
    }
}
```

7.6 res/assets

assets 目录主要用来存放较大的资源，例如 mp3、图片等。这些资源不会在 R 文件中生成对应的 id，assets 里面的文件只能读不能写。目录下可以有目录结构，可以自建目录。

通过"getAssets().list("");"来获取 assets 目录下的所有文件夹和文件的名称，再通过这些名称读取我们想要的文件。

读取文件的方式如下：

```
AssetManager am= null;
am= getAssets();
InputStream is= am.open("filename");
```

举例说明如下：

```
private AssetManager manger;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        info= (TextView)findViewById(R.id.info);
        iv= (ImageView)findViewById(R.id.iv);
        manger= getAssets();
        try {
        //调用读取 Assets 中文本文件内容的方法
         info.setText(getInfoTxt("info.txt"));
         //调用读取 Assets 中图片文件的方法
             iv.setImageBitmap(getImage("pic.PNG"));
           //关闭 manager 对象
             manger.close();
        …
}
}
    //读取文本文件
    public String getInfoTxt(String fileName) throws Exception{
    InputStream is= manger.open(fileName);
    byte[] buffer= new byte[1024];
    int length= - 1;
    StringBuffer s= new StringBuffer();
    while((length= is.read(buffer))!= - 1){
        s.append(new String(buffer, 0, length,"utf- 8"));
              }
            is.close();
            return s.toString();
    }
    //读取图片文件
    public Bitmap getImage(String fileName) throws Exception{
    InputStream is=   manger.open(fileName);
    Bitmap image= BitmapFactory.decodeStream(is)
    is.close();
```

```
        return image;
    }
}
```

7.7 资源的国际化

发布程序的时候,要考虑不同国家的不同用户的语言和习惯不同。

通过使用特殊的技术,让程序界面中的语言根据当前 Android 系统的语言环境变化而自动改变的过程称为国际化。

要实现国际化,只需为不同的环境提供不同语言的资源,并放入到相应的资源文件中,程序运行时,会检测当前的语言环境,根据语言环境再决定读取哪个资源。检测的任务由 Android 系统负责完成。

要包含不同的资源,需要在同一目录下创建并行的文件夹,在每个文件夹后加上合适的名字,这个名字能表明一些配置信息(如语言、原始屏幕等),Android 支持不同类型的修饰语,并可以在文件夹的后面加多条修饰语,修饰语之间以破折号分开。更典型的,可以仅仅指定部分特定的配置选项,只要保证所有的数值都是按顺序排列。资源修饰语如表 7-2 所示。

表 7-2 资源修饰语

修 饰 语	值
语言	两个小写字母 ISO 639-1。例如 en、fr、es
地区	两个大写字母加上一个小写字母'r' ISO 3166-1-alpha-2。例如 rUS、rFR、rES
屏幕方向	port、land、square
屏幕像素	92dpi、108dpi 等
触摸屏类型	notouch、stylus、finger
键盘是否有效	keysexposed、keyshidden
基本文本输入模式	nokeys、qwerty、12key
无触摸屏的主要导航模式	notouch、dpad、trackball、wheel
屏幕分辨率	320×240、640×480 等。大分辨率需要开始指定

下面是一些通用的关于资源目录的命名指导:

(1) 各个变量用破折号分开(每个基本的目录名后跟一个破折号)。

(2) 变量大小写敏感(其大小写法必须始终一致)。例如:

- 一个 drawable 的目录必须命名为 drawable-port,而不是 drawable-PORT。
- 不能有两个目录命名为 drawable-port 以及 drawable-PORT,甚至故意将“port”和“PORT”指为不同的参数也不可以。

(3) 一个式子里同一个类型修饰语中只有一个值是有效的(不能指定 drawable-rEN-

rFR/这样的类型修饰语)。

(4) 可以指定多个参数定义不同的配置,但是参数必须是上面表格里的。例如,drawable-en-rUS-land 的意思是在 US-English 的机器中载入风景视图。

(5) Android 会寻找最适合当前配置的目录,这会在下面描述。

(6) 表格里所列的参数用来打破平衡以防止多重路径限制(看下面的例子)。

(7) 所有目录,无论是限制的,还是不限制的,只要在 res/目录下,都是不能嵌套的(这样 res/drawable/drawable-en 是不可以的)。

所有的资源在被代码引用中最好都使用简单的、不加修饰的名字,如果一个资源这样命名:

```
MyApp/res/drawable-port-92dp/myimage.png
```

它将这样被引用:

```
R.drawable.myimage(code)
@drawable/myimage(XML)
```

只需建立相应的目录和制作相应的资源文件即可。

存放不同的字符资源文件的目录命名规则:

```
资源目录-语言代码-r国家代码
```

资源目录指的是 res 中的子目录,例如 values。

语言代码、国家代码是对应的国际化配置选项,例如,中国为 zh-rCN,美国为 en-rUS。具体定义规则可参看 SDK 中 locale 的定义。

下面是一个例子:

在 res 中分别建立 values-zh-rCN 和 values-en-rUS 两个文件夹,分别存放中文和英文环境下的字符串资源。在两个目录下面分别创建 strings.xml 文件。

在 res 中分别建立 drawable-zh-rCN 和 drawable-en-rUS 两个文件夹。在两个目录下面分别存放中国和美国的国旗图片。

values-zh_rCN 中的 strings.xml 内容如下:

```
<string name="name">姓名</string>
<string name="age">年龄</string>
```

values-en_rUS 中的 strings.xml 内容如下:

```
<string name="name">Name</string>
<string name="age">Age</string>
```

引用资源文件的 layout 的布局文件如下:

```
<TextView
        Android:id="@+id/name"
        Android:layout_width="fill_parent"
        Android:layout_height="wrap_content"
```

```
        Android:text= "@ string/name"
        Android:textSize= "30sp" />
    < TextView
        Android:id= "@ + id/age"
        Android:layout_width= "fill_parent"
        Android:layout_height= "wrap_content"
          Android:text= "@ string/age"
        Android:textSize= "30sp" />
    < ImageView
        Android:id= "@ + id/flag"
        Android:layout_width= "fill_parent"
      Android:src= "@ drawable/flag"
        Android:layout_height= "wrap_content" />
```

第 8 章

Android 菜单

菜单是用户界面中最常见的元素之一，使用非常频繁。菜单是应用程序中非常重要的组成部分，能够在不占用界面空间的前提下，为应用程序提供统一的功能和设置界面，并为程序开发人员提供易于使用的编程接口。

在 Android 中，菜单被分为以下 3 种：选项菜单（OptionsMenu）、上下文菜单（ContextMenu）和子菜单（SubMenu）。

看下面的 3 个界面。第一个界面是按 Menu 按钮弹出的菜单，如图 8.1 所示。



图 8.1　选项菜单

第二个界面是单击第一个界面上的 More 菜单显示的，如图 8.2 所示。

第三个界面是单击第二个界面上的“文件”菜单显示的，如图 8.3 所示。

长按“请单击 Menu 按钮显示选项菜单”文本框，弹出菜单，如图 8.4 所示。

单击“红色背景”，效果如图 8.5 所示。

学习本章之后我们要能自己写出类似这 5 个界面的 Android 菜单程序。

图 8.2　子菜单(1)



图 8.3　子菜单(2)



图 8.4　上下文菜单



图 8.5　菜单功能

8.1　选 项 菜 单

当用户单击设备上的菜单按钮(Menu)，触发事件弹出的菜单就是选项菜单，而用户再次按下 MENU 按钮或返回按钮或者触击屏幕空白处就会关闭选项菜单。选项菜单最多只有 6 个，如果超过 6 个，第 6 个就会自动显示“更多”/More 选项来展示。单击“更多”或 More 显示其余菜单。不同的程序有不同的操作方式和选项菜单，程序在不同的状态下也会有不同的选项菜单。

选项菜单有两种模式：图标模式和扩展模式。

选项菜单中每一个菜单项可以拥有一个图标和一个快捷键(适合带全键盘的设备)。

选项菜单适用于全局活动(类似于 PC 上的进程)，或者用来启动其他活动，而不适用(文字)内容中的选中项(文字内容不能使用选项菜单)。

选项菜单创建方法如下：

(1) 覆盖 Activity 的 onCreateOptionsMenu(Menu menu)方法，此方法当第一次打开菜单时调用。在方法中调用 Menu 的 add()方法添加菜单项(MenuItem)，可以调用 MenuItem 的 setIcon()方法为菜单项设置图标。也可以在资源中配置。最后返回 true，如果是 false，菜单则不会显示。

(2) 为菜单项注册事件。覆盖 Activity 的 onOptionsMenuSelected()方法响应菜单项(MenuItem)选中事件。

(3) 如果需要有菜单关闭之后的动作的,覆盖 Activity 的 onOptionsMenuClosed(Menu menu)。

(4) 如果需要在选项菜单显示之前调整菜单内容的,覆盖 Activity 的 onPrepareOptionsMenu(Menu menu)。

(5) 如果需要有菜单打开之后的动作的,覆盖 Activity 的 onMenuOpened(int featureId, Menu menu)。

下面举例说明。

1. 默认模式

单击 Menu 按钮打开时,显示效果如图 8.6 所示。



图 8.6 选项菜单(1)

再次单击 Menu 按钮关闭时,如图 8.7 所示。



图 8.7 选项菜单(2)

界面的布局 XML 文件如下:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
Android:orientation="vertical" Android:layout_width="fill_parent"
Android:layout_height="fill_parent" >
<TextView Android:layout_width="wrap_content"
Android:layout_height="wrap_content" Android:text="请单击 Menu按钮显示选项菜单"
Android:id="@+id/TextView02" />
</LinearLayout>
```

对应的 Activity 的 Java 代码如下:

```
import Android.app.Activity;
import Android.os.Bundle;
```

```
import Android.view.Menu;
import Android.view.MenuItem;
import Android.widget.Toast;
public class MenuActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.menu);
    }
    public boolean onCreateOptionsMenu(Menu menu) {
        menu.add(Menu.NONE, Menu.FIRST+ 1, 5, "删除").setIcon(
                Android.R.drawable.ic_menu_delete);
        menu.add(Menu.NONE, Menu.FIRST+ 2, 2, "保存").setIcon(
                Android.R.drawable.ic_menu_edit);
        menu.add(Menu.NONE, Menu.FIRST+ 3, 6, "帮助").setIcon(
                Android.R.drawable.ic_menu_help);
        menu.add(Menu.NONE, Menu.FIRST+ 4, 1, "添加").setIcon(
                Android.R.drawable.ic_menu_add);
        menu.add(Menu.NONE, Menu.FIRST+ 5, 4, "详细").setIcon(
                Android.R.drawable.ic_menu_info_details);
        menu.add(Menu.NONE, Menu.FIRST+ 6, 3, "发送").setIcon(
                Android.R.drawable.ic_menu_send);
        return true;
    }
    public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()) {
        case Menu.FIRST+ 1:
            Toast.makeText(this, "ItemId= "+ item.getItemId()
                                 + "--删除菜单被单击了",
                              Toast.LENGTH_LONG).show();
            break;
        case Menu.FIRST+ 2:
            Toast.makeText(this, "ItemId= "+ item.getItemId()
                               + "--保存菜单被单击了",
                               Toast.LENGTH_LONG).show();
            break;
        case Menu.FIRST+ 3:
            Toast.makeText(this, "ItemId= "+ item.getItemId()
                               + "--帮助菜单被单击了",
                               Toast.LENGTH_LONG).show();
            break;
        case Menu.FIRST+ 4:
            Toast.makeText(this, "ItemId= "+ item.getItemId()
                               + "--添加菜单被单击了",
                               Toast.LENGTH_LONG).show();
```

```
                break;
            case Menu.FIRST+5:
                Toast.makeText(this, "ItemId="+item.getItemId()
                                        +"--详细菜单被单击了",
                                        Toast.LENGTH_LONG).show();
                break;
            case Menu.FIRST+6:
                Toast.makeText(this, "ItemId="+item.getItemId()
                                        +"--发送菜单被单击了",
                                        Toast.LENGTH_LONG).show();
                break;
            }
            return false;
        }
        public void onOptionsMenuClosed(Menu menu) {
            Toast.makeText(this, "选项菜单关闭了", Toast.LENGTH_LONG).show();
        }
        public boolean onPrepareOptionsMenu(Menu menu) {
        Toast.makeText(this, "选项菜单显示之前 onPrepareOptionsMenu 方法会被调用",
                    Toast.LENGTH_LONG).show();
            //如果返回 false,此方法就把用户单击 menu 的动作屏蔽了,
            //onCreateOptionsMenu 方法将不会被调用
            return true;
        }
}
```

2. 自定义模式

单击 Menu 按钮打开时,显示效果如图 8.8 所示。

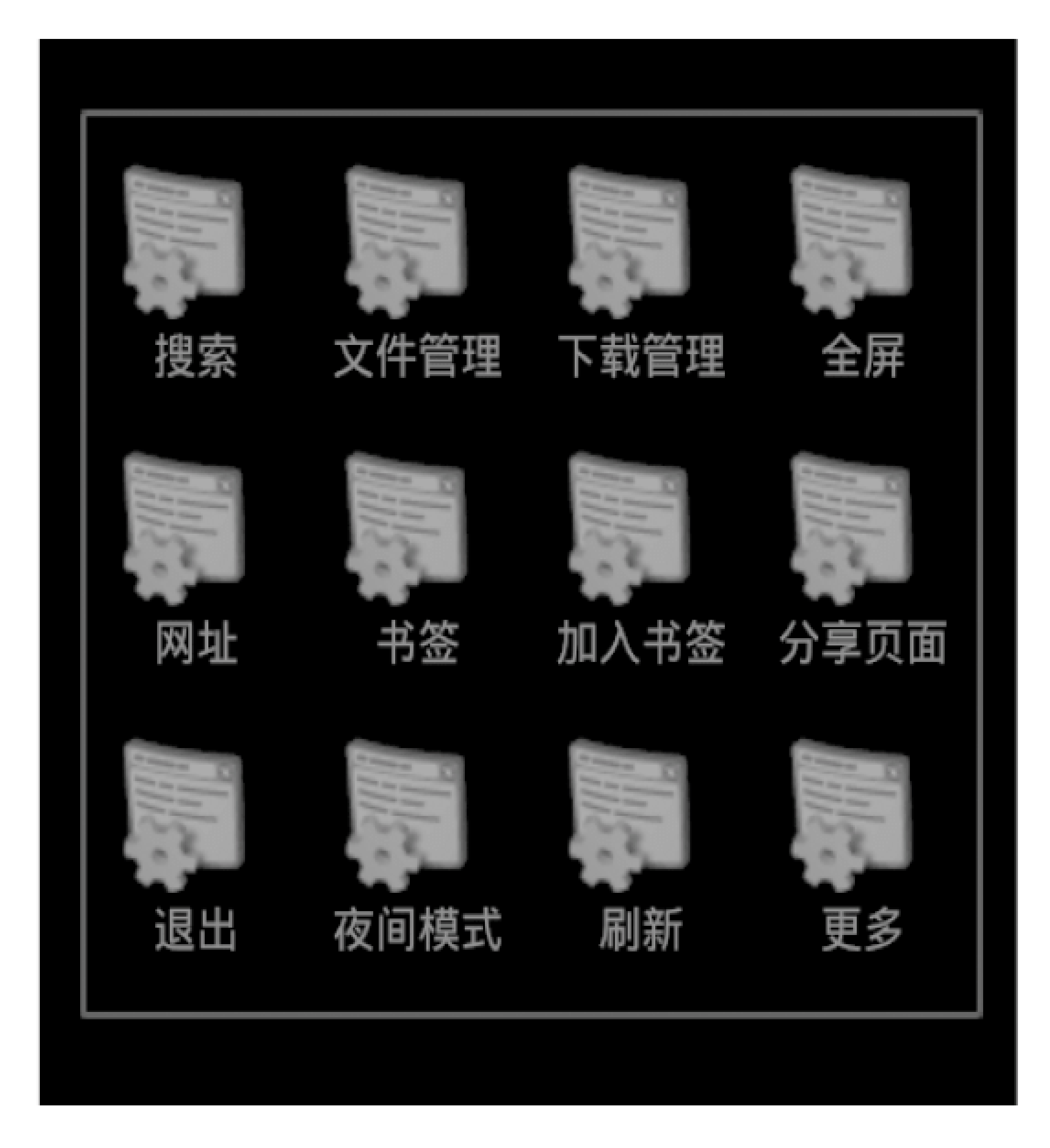


图 8.8 自定义效果

单击“更多”菜单时，显示效果如图 8.9 所示。

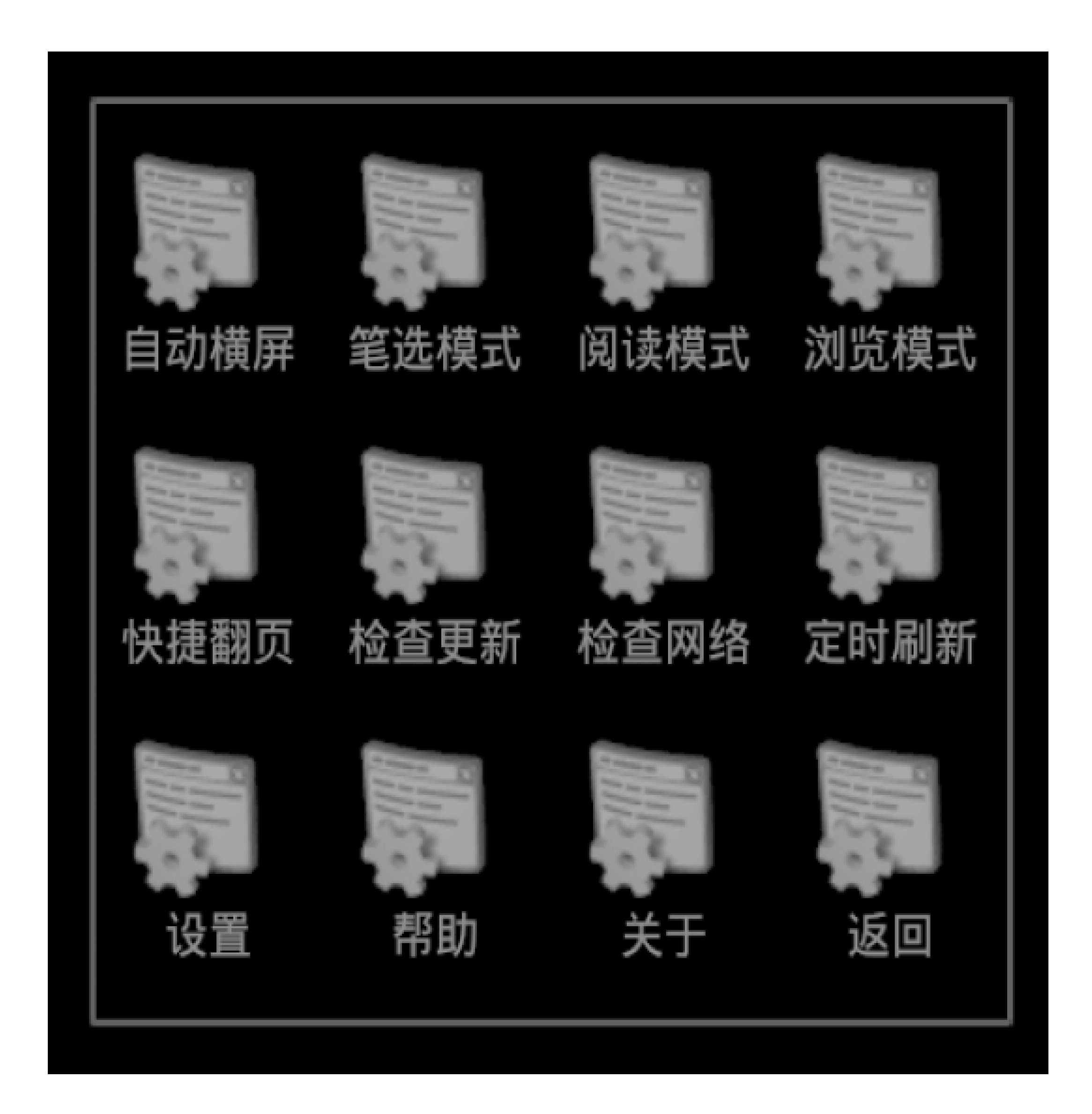


图 8.9　自定义效果

单击“返回”菜单，返回图 8.8。

界面的布局文件与默认模式类似。

界面的菜单显示列表的布局 XML 文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:layout_width="fill_parent"
    Android:layout_height="fill_parent"
    Android:orientation="vertical" >
    <GridView
        Android:id="@+id/gridview"
        Android:layout_width="fill_parent"
        Android:layout_height="fill_parent"
        Android:gravity="center"
        Android:horizontalSpacing="10dip"
        Android:numColumns="4"
        Android:stretchMode="columnWidth"
        Android:verticalSpacing="10dip" />
</LinearLayout>
```

界面的菜单项的布局文件 XML 代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:Android=
          "http://schemas.Android.com/apk/res/Android"
Android:id="@+id/RelativeLayout_Item"
Android:layout_width="fill_parent" Android:layout_height="wrap_content"
```

```
Android:paddingBottom= "5dip">
<ImageView Android:id= "@+id/item_image"
Android:layout_centerHorizontal= "true"
Android:layout_width= "wrap_content"
Android:layout_height= "wrap_content"></ImageView>
<TextView Android:layout_below= "@ id/item_image"
Android:id= "@+id/item_text"
Android:layout_centerHorizontal= "true"
Android:layout_width= "wrap_content"
Android:layout_height= "wrap_content" Android:text= "选项"></TextView>
</RelativeLayout>
```

对应的 Activity 的 Java 代码如下：

```
import Java.util.ArrayList;
import Java.util.HashMap;
import Android.app.Activity;
import Android.app.AlertDialog;
import Android.content.DialogInterface;
import Android.content.DialogInterface.OnKeyListener;
import Android.os.Bundle;
import Android.view.KeyEvent;
import Android.view.Menu;
import Android.view.View;
import Android.widget.AdapterView;
import Android.widget.GridView;
import Android.widget.SimpleAdapter;
import Android.widget.AdapterView.OnItemClickListener;
public class MenuActivity extends Activity {
    private final int ITEM_SEARCH= 0;            //搜索
    private final int ITEM_FILE_MANAGER= 1;      //文件管理
    private final int ITEM_DOWN_MANAGER= 2;      //下载管理
    private final int ITEM_FULLSCREEN= 3;        //全屏
    private final int ITEM_MORE= 11;             //菜单
    private boolean isMore= false;               //menu菜单翻页控制
    AlertDialog menuDialog;                      //menu菜单 Dialog
    GridView menuGrid;
    View menuView;
    /** 菜单图片 **/
    int[] menu_image_array= {R.drawable.icon, R.drawable.icon,
            R.drawable.icon, R.drawable.icon, R.drawable.icon,
            R.drawable.icon,R.drawable.icon, R.drawable.icon,
            R.drawable.icon, R.drawable.icon,R.drawable.icon,
            R.drawable.icon };
    /** 菜单文字 **/
```

```
String[] menu_name_array= {"搜索", "文件管理", "下载管理", "全屏",
                          "网址", "书签","加入书签", "分享页面", "退出",
                          "夜间模式", "刷新", "更多"};
/** 菜单图片 2 **/
int[] menu_image_array2= {R.drawable.icon, R.drawable.icon,
        R.drawable.icon, R.drawable.icon, R.drawable.icon,
        R.drawable.icon,R.drawable.icon, R.drawable.icon,
        R.drawable.icon, R.drawable.icon,R.drawable.icon,
        R.drawable.icon };
/** 菜单文字 2 **/
String[] menu_name_array2= {"自动横屏", "笔选模式", "阅读模式",
                           "浏览模式", "快捷翻页","检查更新", "检查网络",
                           "定时刷新", "设置", "帮助", "关于", "返回"};
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.menua);
    menuView= View.inflate(this, R.layout.menugrid, null);
    //创建 AlertDialog
    menuDialog= new AlertDialog.Builder(this).create();
    menuDialog.setView(menuView);
    menuDialog.setOnKeyListener(new OnKeyListener() {
        public boolean onKey(DialogInterface dialog, int keyCode,
                KeyEvent event) {
            if(keyCode= = KeyEvent.KEYCODE_MENU)        //监听按键
                dialog.dismiss();
            return false;
        }
    });
    menuGrid= (GridView) menuView.findViewById(R.id.gridview);
    menuGrid.setAdapter(getMenuAdapter(menu_name_array,
                menu_image_array));
    /** 监听 menu 选项 **/
    menuGrid.setOnItemClickListener(new OnItemClickListener() {
        public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                long arg3) {
            switch(arg2) {
            case ITEM_SEARCH:                //搜索
                break;
            case ITEM_FILE_MANAGER:          //文件管理
                break;
            case ITEM_DOWN_MANAGER:          //下载管理
                break;
            case ITEM_FULLSCREEN:            //全屏
```

```
                    break;
                case ITEM_MORE:                     //翻页
                    if(isMore) {
                        menuGrid.setAdapter(
                                getMenuAdapter(menu_name_array2,
                                menu_image_array2));
                        isMore= false;
                    } else {                        //首页
                        menuGrid.setAdapter(
                                getMenuAdapter(menu_name_array,
                                menu_image_array));
                        isMore= true;
                    }
                    menuGrid.invalidate();          //更新 menu
                    menuGrid.setSelection(ITEM_MORE);
                    break;
                }
            }
        });
    }
    public boolean onCreateOptionsMenu(Menu menu) {
        menu.add("menu");                           //必须创建一项
        return super.onCreateOptionsMenu(menu);
    }
    private SimpleAdapter getMenuAdapter(String[] menuNameArray,
            int[] imageResourceArray) {
        ArrayList<HashMap<String, Object>> data=
                new ArrayList<HashMap<String, Object>>();
        for(int i=0; i<menuNameArray.length; i++) {
            HashMap<String, Object> map= new HashMap<String, Object>();
            map.put("itemImage", imageResourceArray[i]);
            map.put("itemText", menuNameArray[i]);
            data.add(map);
        }
        SimpleAdapter simperAdapter= new SimpleAdapter(this, data,
                R.layout.menuitem, new String[] {"itemImage", "itemText"},
                new int[] {R.id.item_image, R.id.item_text});
        return simperAdapter;
    }
    public boolean onMenuOpened(int featureId, Menu menu) {
        if(menuDialog== null) {
            menuDialog=
                    new AlertDialog.Builder(this).setView(menuView).show();
```

```
            } else {
                menuDialog.show();
            }
            return false;                          //返回 true 则显示系统 menu
        }
    }
```

3. 使用资源设置菜单

参见 7.4 节的 res/menu 的内容。

8.2 子　菜　单

子菜单就是将相同功能的分组进行多级显示的一种菜单，比如，Windows 的“文件”菜单中就有“新建”、“打开”、“关闭”等子菜单。子菜单是一种自然的组织菜单项的方式，它被大量地运用在 Windows 和其他 OS 的 GUI 设计中。Android 同样支持子菜单，可以通过 addSubMenu(int groupId, int itemId, int order, int titleRes)方法非常方便地创建和响应子菜单。

在 Android 中单击子菜单将弹出悬浮窗口显示子菜单项。子菜单不支持嵌套，即子菜单中不能再包括其他子菜单。可以为子菜单添加图标，但是不会显示其菜单项的图标，这一点需要留意。除了代码中的 setIcon(int iconRes)方法，还有一个 setHeaderIcon(int iconRes)方法可以添加子菜单项栏目的标题图标。

一个子菜单是一个在已有菜单的某个菜单项上打开的菜单。可以向任何菜单添加子菜单。当程序拥有很多功能并可按类别组织起来时，子菜单是最佳选择。

创建子菜单的方法同创建选项菜单相似：

(1) 覆盖 Activity 的 onCreateOptionsMenu()方法，调用 Menu 的 addSubMenu()方法添加菜单项。调用 SubMenu 的 add()方法添加子菜单项。

(2) 为菜单项注册事件。覆盖 onCreateItemSelected()方法，响应菜单单击事件。

下面举例说明。

1. 单选按钮显示

单击 Menu 按钮打开时，显示效果如图 8.10 所示。



图 8.10　子菜单

单击“搜索”菜单，显示效果如图 8.11 所示。

界面的布局文件与默认模式类似。

对应的 Activity 的 Java 代码如下：



图 8.11 子菜单菜单项

```
import Android.os.Bundle;
import Android.app.Activity;
import Android.view.Menu;
import Android.view.MenuItem;
import Android.view.SubMenu;
import Android.widget.Toast;
public class MenuBActivity extends Activity {
    public static final int MENU_LOCAL= 0;
    public static final int MENU_INTERENT= 1;
    public static final int MENU_SEARCH= 3;
    MenuItem local_MenuItem= null;
    MenuItem internet_MenuItem= null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.menub);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        SubMenu sub= menu.addSubMenu("搜索");
        sub.setIcon(Android.R.drawable.ic_menu_search);
        local_MenuItem= sub.add(0, MENU_LOCAL, 0, "本地");
        internet_MenuItem= sub.add(0, MENU_INTERENT, 0, "网络");
        local_MenuItem.setCheckable(true);
        //将 local_menuItem菜单项设置为已选
        local_MenuItem.setChecked(true);
        //设置菜单项为单选菜单项,互斥的
        sub.setGroupCheckable(0, true, true);
        SubMenu sub2= menu.addSubMenu("记录");
        return true;
```

```
    }
    public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()) {
        case MENU_LOCAL:
            setTitle("本地搜索");
            Toast.makeText(this, "Search Local", 0).show();
            break;
        case MENU_INTERENT:
            setTitle("网络搜索");
            Toast.makeText(this, "Search Internet", 0).show();
            break;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

2. 复选框显示

单击“搜索”菜单，显示效果如图 8.12 所示。



图 8.12 子菜单菜单项

只需修改 onCreateOptionsMenu 方法，代码如下：

```
super.onCreateOptionsMenu(menu);
    SubMenu sub= menu.addSubMenu("搜索");
    sub.setIcon(Android.R.drawable.ic_menu_search);
    local_MenuItem= sub.add(0,MENU_LOCAL,0,"本地");
    internet_MenuItem= sub.add(0,MENU_INTERENT,0,"网络");
    //将菜单项设置为复选
    local_MenuItem.setCheckable(true);
    internet_MenuItem.setCheckable(true);
    local_MenuItem.setChecked(true);
    SubMenu sub2= menu.addSubMenu("记录");
return true;
```

3. 使用资源

单击 Menu 按钮打开时,显示效果如图 8.13 所示。



图 8.13　子菜单

单击"菜单一",显示效果如图 8.14 所示。



图 8.14　子菜单菜单项

界面的布局文件与默认模式类似。

资源文件 res/menu/menus.xml 如下:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:Android="http://schemas.Android.com/apk/res/Android">
    <item Android:id="@+id/item1" Android:icon="@drawable/icon"
                        Android:title="菜单一">
      <menu>
         <!--menu group-->
         <group Android:id="@+id/group1"
                        Android:checkableBehavior="single">
            <item Android:id="@+id/groupItem1" Android:title="组 1"/>
```

```
                <item Android:id="@+id/groupItem2" Android:title="组 2"/>
                <item Android:id="@+id/groupItem3" Android:title="组 3"/>
                <item Android:id="@+id/groupItem4" Android:title="组 4"/>
                <item Android:id="@+id/groupItem5" Android:title="组 5"/>
                <item Android:id="@+id/groupItem6" Android:title="组 6"/>
                <item Android:id="@+id/groupItem7" Android:title="组 7"/>
                <item Android:id="@+id/groupItem8" Android:checked="true"
                                     Android:title="组 8"/>
            </group>
        </menu>
    </item>
    <item Android:id="@+id/item2" Android:title="菜单二"></item>
</menu>
```

对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.Menu;
import Android.view.MenuInflater;
import Android.view.MenuItem;
import Android.widget.Toast;
public class MenuEActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.menu);
    }
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater=getMenuInflater();
        inflater.inflate(R.menu.menus, menu);
        return true;
    }
    public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()) {
        case R.id.groupItem1:
            Toast.makeText(this, "create new", 0).show();
            break;
        case R.id.groupItem2:
            Toast.makeText(this, "create new", 0).show();
            break;
        case R.id.groupItem3:
            Toast.makeText(this, "create new", 0).show();
            break;
        case R.id.groupItem4:
            Toast.makeText(this, "create new", 0).show();
```

```
                break;
            case R.id.groupItem5:
                Toast.makeText(this, "create new", 0).show();
                break;
            case R.id.groupItem6:
                Toast.makeText(this, "create new", 0).show();
                break;
            case R.id.groupItem7:
                Toast.makeText(this, "create new", 0).show();
                break;
            case R.id.groupItem8:
                if(item.isChecked())
                    item.setChecked(false);
                else
                    item.setChecked(true);
                break;
            default:
                return super.onOptionsItemSelected(item);
            }
            return true;
        }
    }
```

8.3　上下文菜单

在 Windows 中，我们已经习惯了右击文件来执行“打开”、“重命名”、“剪切”、“删除”等操作，这个右键弹出的菜单就是上下文菜单。Android 的上下文菜单项是不能用快捷键的。手机的操作方式与使用鼠标的 PC 操作方式不同，Android 是通过长按某个视图元素来弹出上下文菜单的。

使用文本菜单被认为是快速执行常规操作的一种捷径，文本菜单比起某些常显示的按钮或选项菜单，出现的机会更少。很多用户从未发现或者使用过文本菜单。正因为如此，文本菜单上的每个指令也应该在界面上利用多种形式（比如图标、按钮之类）直观地显示。

上下文菜单继承了 Android.view.Menu，因此我们可以像操作 Options Menu 那样给上下文菜单增加菜单项。上下文菜单与 Options Menu 最大的不同在于，Options Menu 的拥有者是 Activity，而上下文菜单的拥有者是 Activity 中的 View。每个 Activity 有且只有一个 Options Menu，它为整个 Activity 服务。而一个 Activity 往往有多个 View，并不是每个 View 都有上下文菜单，这就需要我们显式地通过 registerForContextMenu (View view)来指定。

创建上下文菜单的方法如下：

(1) 覆盖 Activity 的 onCreateContextMenu()方法，调用 Menu 的 add 方法添加菜单

项 MenuItem。每调用一次,onCreateContextMenu 就被调用一次。

(2) 覆盖 onContextItemSelected()方法,响应菜单单击事件。

(3) 调用 registerForContextMenu()方法,为视图注册上下文菜单。

下面举例说明。

长按 ListView 的某一行时,显示效果如图 8.15 所示。



图 8.15 上下文菜单

界面布局的 XML 文件如下:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:layout_width="fill_parent"
    Android:layout_height="fill_parent"
    Android:orientation="vertical" >
    <ListView  Android:id="@+id/listView1"
        Android:layout_width="fill_parent"
        Android:layout_height="wrap_content" >
  </ListView>
</LinearLayout>
```

对应的 Activity 的 Java 代码如下:

```
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.ContextMenu;
import Android.view.ContextMenu.ContextMenuInfo;
import Android.view.Menu;
import Android.view.MenuItem;
import Android.view.View;
import Android.view.ViewGroup;
```

```
import Android.widget.BaseAdapter;
import Android.widget.ListView;
import Android.widget.TextView;
import Android.widget.Toast;
public class MenuFActivity extends Activity {
    private ListView listView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.menulist);
        listView= (ListView) this.findViewById(R.id.listView1);
        listView.setAdapter(new MyAdapter());
        this.registerForContextMenu(listView);
    }
    @Override
    public void onCreateContextMenu(ContextMenu menu, View v,
            ContextMenuInfo menuInfo) {
        //创建菜单
        super.onCreateContextMenu(menu, v, menuInfo);
        menu.setHeaderTitle("文件操作");
        menu.add(1, Menu.FIRST, Menu.NONE, "发送");
        menu.add(1, Menu.FIRST+ 1, Menu.NONE, "标记为重要");
        menu.add(1, Menu.FIRST+ 2, Menu.NONE, "重命名");
        menu.add(1, Menu.FIRST+ 3, Menu.NONE, "删除");
    }
    @Override
    public boolean onContextItemSelected(MenuItem item) {
        //菜单事件
        switch(item.getItemId()) {
        case 1:
            Toast.makeText(this, "发送", 1).show();
            break;
        case 2:
            Toast.makeText(this, "标记为重要", 1).show();
            break;
        case 3:
            Toast.makeText(this, "重命名", 1).show();
            break;
        case 4:
            Toast.makeText(this, "删除", 1).show();
            break;
        default:
            return super.onContextItemSelected(item);
        }
```

```
            return true;
        }
        //给 ListView提供一些数据
        private class MyAdapter extends BaseAdapter {
            public int getCount() {
                return 5;
            }
            public Object getItem(int position) {
                return null;
            }
            public long getItemId(int position) {
                return 0;
            }
        public View getView(int position, View convertView, ViewGroup parent) {
                TextView tv= new TextView(MenuFActivity.this);
                tv.setText("文件"+ position);
                return tv;
            }
        }
}
```

8.4 实例代码

下面给出本章开始给出的几个菜单的实现代码。

界面布局的 XML 文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
Android:orientation="vertical" Android:layout_width="fill_parent"
Android:layout_height="fill_parent">
<TextView Android:layout_width="wrap_content"
Android:layout_height="wrap_content" Android:text="请单击 Menu键显示选项菜单"
Android:id="@+id/mytextview" />
</LinearLayout>
```

对应的 Activity 的 Java 代码如下：

```
import Android.app.Activity;
import Android.graphics.Color;
import Android.os.Bundle;
import Android.view.ContextMenu;
import Android.view.ContextMenu.ContextMenuInfo;
import Android.view.Menu;
import Android.view.MenuItem;
```

```
import Android.view.SubMenu;
import Android.view.View;
import Android.widget.TextView;
import Android.widget.Toast;
public class MenuGActivity extends Activity {
    //菜单项 ID FIRST为 Menu类中的整型常量
    private static final int ITEM1= Menu.FIRST;
    private static final int ITEM2= Menu.FIRST+ 1;
    private static final int ITEM3= Menu.FIRST+ 2;
    private static final int ITEM4= Menu.FIRST+ 3;
    private static final int ITEM5= Menu.FIRST+ 4;
    private static final int ITEM6= Menu.FIRST+ 5;
    private static final int ITEM7= Menu.FIRST+ 6;
    private static final int ITEM8= Menu.FIRST+ 7;
    private static final int ITEM9= Menu.FIRST+ 8;
    private static final int ITEM10= Menu.FIRST+ 9;
    TextView myTextView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.menug);
        myTextView= (TextView) findViewById(R.id.mytextview);
        registerForContextMenu(myTextView);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //选项菜单
        menu.add(0, ITEM1, 0, "开始").setIcon(R.drawable.icon);
        menu.add(0, ITEM2, 0, "开始 1");
        menu.add(1, ITEM3, 0, "开始 2");
        menu.add(1, ITEM4, 0, "开始 3");
        menu.add(1, ITEM5, 0, "开始 4");
        //添加子菜单
        SubMenu subFile= menu.addSubMenu("文件");
        SubMenu editFile= menu.addSubMenu("编辑");
        //为子菜单添加菜单项
        subFile.add(0, ITEM6, 0, "新建");
        subFile.add(0, ITEM7, 0, "打开");
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()) {
```

```
        case ITEM1:
            Toast.makeText(this, "开始", 1).show();
            break;
        case ITEM2:
            Toast.makeText(this, "开始 1", 1).show();
            break;
        case ITEM3:
            Toast.makeText(this, "开始 2", 1).show();
            break;
        case ITEM4:
            Toast.makeText(this, "开始 3", 1).show();
            break;
        case ITEM5:
            Toast.makeText(this, "开始 4", 1).show();
            break;
        case ITEM6:
            Toast.makeText(this, "新建文件", 1).show();
            break;
        case ITEM7:
            Toast.makeText(this, "打开文件", 1).show();
            break;
        }
        return true;
    }
    @Override
    public void onCreateContextMenu(ContextMenu menu, View v,
            ContextMenuInfo menuInfo) {
        //向上下文菜单中添加菜单项,注意此处的 menu 是 ContextMenu
        menu.add(0, ITEM8, 0, "红色背景");
        menu.add(0, ITEM9, 0, "绿色背景");
        menu.add(1, ITEM10, 0, "白色背景");
    }
    @Override
    public boolean onContextItemSelected(MenuItem item) {
        switch(item.getItemId()) {
        case ITEM8:
            myTextView.setBackgroundColor(Color.RED);
            break;
        case ITEM9:
            myTextView.setBackgroundColor(Color.GREEN);
            break;
        case ITEM10:
            myTextView.setBackgroundColor(Color.WHITE);
```

```
                break;
            }
            return true;
        }
    }
```

第 9 章 数据存储

数据存储在开发中是使用最频繁的。我们在 Java 开发中可以使用数据库、一般文件、参数文件存储数据。在 Android 平台也是类似的。

Android 提供了 5 种方式实现数据存储，分别是使用 SharedPreferences 存储数据、文件存储数据、SQLite 数据库存储数据、使用 ContentProvider 存储数据和网络存储数据。

9.1 使用 SharedPreferences 存储数据

SharedPreferences 是 Android 平台上一个轻量级的存储类，主要保存一些常用的配置，例如窗口状态，一般在 Activity 中，重载窗口状态 onSaveInstanceState 的保存一般使用 SharedPreferences 完成，它提供了 Android 平台常规的 Long 长整型、Int 整型、String 字符串型的保存。事实上，它完全相当于一个 HashMap，唯一不同的是 HashMap 中的 Value 可以是任何对象，而 SharedPreferences 中的值只能存储基本数据类型（primitive types）。

它的本质是基于 XML 文件存储 key-value 键值对数据，通常用来存储一些简单的配置信息。在 Android 中 SharedPreferences 使用最多的地方是用来保存配置（Settings）信息，系统中的 Settings 是这样，各个应用中的 Settings 也是这样。并且，在 Android 中为了方便地使用 SharedPreferences 保存配置信息，专门有 PreferenceActivity 用来封装。也就是说，如果想在应用程序中创建配置（Settings），可以直接使用 PreferenceActivity 和一些相关的专门为 Preference 封装的组件，而不用再直接去创建，读取和保存 SharedPreference，Framework 中的这些组件会做这些工作。

其存储位置在/data/data/<应用程序主包>/shared_prefs 目录下。

SharedPreferences 对象本身只能获取数据而不支持存储和修改，存储和修改是通过 Editor 对象实现。

实现 SharedPreferences 存储的步骤如下：

（1）根据 Context 获取 SharedPreferences 对象。

（2）利用 edit()方法获取 Editor 对象。

（3）通过 Editor 对象存储 key-value 键值对数据。

（4）通过 commit()方法提交数据。

9.1.1 获得 SharedPreferences

```
SharedPreferences getSharedPreferences(String name, int mode)
```

参数：

name：xml 文件名字，后缀会主动加上.xml。

mode：操作 xml 的模式，有以下几种：

(1) 0 或者 Context.MODE_PRIVATE：默认操作，表示只有创建文件的程序对该目标可读可写。

(2) Context.MODE_APPEND：表示追加数据。

(3) Context.MODE_WORLD_READABLE：全局读，除了创建文件的程序对该目标可读可写之外，其他程序可以对该文件进行读操作。

(4) Context.MODE_WORLD_WRITEABLE：全局写，除了创建文件的程序对该目标可读可写之外，其他程序可以对该文件进行写操作。

注意：对于 XML 简单数据存储，Context.MODE_PRIVATE 和 Context.MODE_APPEND 效果是一样的，而后面的 io 文件存储效果则不同。

```
SharedPreferences sp= this.getSharedPreferences("user",
                        Context.MODE_WORLD_READABLE);
```

9.1.2 增加或者更新数据

```
SharedPreferences sp= this.getSharedPreferences("user",
                        Context.MODE_WORLD_READABLE);
//xml 文件编辑器
Editor editor= sp.edit();
editor.putString("user", user);
editor.putString("password", password);
//需要提交
editor.commit();
```

9.1.3 读取数据

```
SharedPreferences sp= this.getSharedPreferences("user",
                        Activity.MODE_PRIVATE);
String user= sp.getString("user", "没有 user 变量");
String password= sp.getString("password", "没有 password 变量");
```

9.1.4 清空数据

```
SharedPreferences sp= this.getSharedPreferences("user",
                        Context.MODE_WORLD_READABLE);
//xml 文件编辑器
```

```
Editor editor= sp.edit();
editor.clear();
//需要提交
editor.commit();
```

下面举例说明。

```
public class MainActivity extends Activity {
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
//获取 SharedPreferences 对象
Context ctx= MainActivity.this;
SharedPreferences sp= ctx.getSharedPreferences("SP", MODE_PRIVATE);
//存入数据
Editor editor= sp.edit();
editor.putString("STRING_KEY", "string");
editor.putInt("INT_KEY", 0);
editor.putBoolean("BOOLEAN_KEY", true);
editor.commit();
//返回 STRING_KEY 的值
Log.d("SP", sp.getString("STRING_KEY", "none"));
//如果 NOT_EXIST 不存在,则返回值为"none"
Log.d("SP", sp.getString("NOT_EXIST", "none"));
}
}
```

这段代码执行过后,即在/data/data/com.test/shared_prefs 目录下生成了一个 SP.xml 文件,一个应用可以创建多个这样的 XML 文件。

9.1.5 PreferenceActivity

PreferenceActivity 是专门用来实现程序设置界面及参数存储的一个 Activity。PreferenceActivity 是 Android 提供的对系统信息和配置进行自动保存的 Activity,它通过 SharedPreference 方式将信息保存在 XML 文件中。使用 PreferenceActivity 不需要对 SharedPreference 进行操作,系统会自动对 Activity 的各种 View 上的改变进行保存。

- EditTextPreference:输入编辑框,值为 String 类型,会弹出对话框供输入。
- Preference:只进行文本显示,需要与其他进行组合使用。
- RingtonePreference:系统铃声选择。
- PreferenceScreen:设置页面,可嵌套形成二级设置页面,用 Title 参数设置标题。
- PreferenceCategory:某一类相关的设置,可用 Title 参数设置标题。
- CheckBoxPreference:是一个 CheckBox 设置,只有两种值:true 或 false,可用 Title 参数设置标题,用 summaryOn 和 summaryOff 参数设置控件选中和未选中时的提示。
- ListPreference:下拉框选择控件,用 Title 参数设置标题,用 Summary 参数设置

说明,单击后出现下拉框,用 dialogTitle 设置下拉框的标题,下拉框内显示的内容和具体的值需要在 res/values/array. xml 中设置两个 array 来表示。

下面举例说明。

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:Android=
                    "http://schemas.Android.com/apk/res/Android">
    <!--类别 -->
    <PreferenceCategory Android:title="复选框组">
        <!--测试复选框 -->
        <CheckBoxPreference
            Android:key="music"
            Android:summary="播放背景音乐"
            Android:title="音乐"/>
        <CheckBoxPreference
            Android:key="hints"
            Android:summary="使用提示"
            Android:summaryOff="关闭提示"
            Android:summaryOn="已经使用提示"
            Android:title="提示"/>
    </PreferenceCategory>
    <PreferenceCategory Android:title="文本框">
        <!--测试编辑框 -->
        <EditTextPreference
            Android:dialogMessage="请输入用户名"
            Android:dialogTitle="用户信息对话框"
            Android:key="user"
            Android:positiveButtonText="确定"
            Android:summary="编写用户名"
            Android:title="编写用户名"/>
    </PreferenceCategory>
    <!--测试列表 -->
    <ListPreference
        Android:entries="@array/entries_list_preference"
        Android:entryValues="@array/entriesvalue_list_preference"
        Android:key="lang"
        Android:negativeButtonText="否"
        Android:positiveButtonText="确定"
        Android:summary="请选择语言"
        Android:title="语言设置"/>
    <PreferenceCategory Android:title="其他选项">
        <RingtonePreference
            Android:key="rt1"
            Android:summary="选择铃声"
```

```
            Android:title="RingtonePreference Sample" />
    </PreferenceCategory>
</PreferenceScreen>
res/values/arrays.xml:
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="entries_list_preference">
        <item>中文</item>
        <item>英文</item>
        <item>日文</item>
    </string-array>
    <string-array name="entriesvalue_list_preference">
        <item>zh</item>
        <item>en</item>
        <item>jn</item>
    </string-array>
</resources>
```

SharedPreferences 对象与 SQLite 数据库相比,免去了创建数据库、创建表、写 SQL 语句等诸多操作,相对而言更加方便、简洁。但是 SharedPreferences 也有其自身缺陷,例如,只能存储 boolean、int、float、long 和 String 5 种简单的数据类型,无法进行条件查询,等等。所以不论 SharedPreferences 的数据存储操作如何简单,它也只能是存储方式的一种补充,而无法完全替代如 SQLite 数据库这样的其他数据存储方式。

9.2 文件存储数据

关于文件存储,Activity 提供了 openFileOutput()方法用来把数据输出到文件中,具体的实现过程与在 J2SE 环境中保存数据到文件中是一样的。

文件可用来存放大量数据,如文本、图片、音频等。

默认位置:/data/data/<应用程序主包>/files/***.***。

代码示例:

```
public void fileopensave()
{
    try {
        FileOutputStream outStream=
                this.openFileOutput("file.txt",Context.MODE_WORLD_READABLE);
        outStream.write(text.getText().toString().getBytes());
        outStream.close();
        Toast.makeText(MyActivity.this,"Saved",Toast.LENGTH_LONG).show();
    } catch(FileNotFoundException e) {
        return;
    }
```

```
        catch(IOException e){
            return;
        }
    }
```

openFileOutput()方法的第一个参数用于指定文件名称，不能包含路径分隔符“/”，如果文件不存在，Android 会自动创建它。

创建的文件保存在/data/data/<package name>/files 目录下，目录结构如下：/data/data/cn.itcast.action/files/itcast.txt。

单击 Eclipse 菜单 Window→Show View→Other，在对话窗口中展开 Android 文件夹，选择下面的 File Explorer 视图，然后在 File Explorer 视图中展开/data/data/<package name>/files 目录就可以看到该文件。

openFileOutput()方法的第二个参数用于指定操作模式，有 4 种模式，分别为：

(1) Context.MODE_PRIVATE=0

(2) Context.MODE_APPEND=32768

(3) Context.MODE_WORLD_READABLE=1

(4) Context.MODE_WORLD_WRITEABLE=2

其中：

(1) Context.MODE_PRIVATE：为默认操作模式，代表该文件是私有数据，只能被应用本身访问，在该模式下，写入的内容会覆盖原文件的内容，如果想把新写入的内容追加到原文件中，可以使用 Context.MODE_APPEND。

(2) Context.MODE_APPEND：模式会检查文件是否存在，存在就往文件追加内容，否则就创建新文件。

(3) Context.MODE_WORLD_READABLE 和 Context.MODE_WORLD_WRITEABLE 用来控制其他应用是否有权限读写该文件。

- MODE_WORLD_READABLE：表示当前文件可以被其他应用读取。
- MODE_WORLD_WRITEABLE：表示当前文件可以被其他应用写入。

如果希望文件被其他应用读和写，可以传入“openFileOutput("itcast.txt", Context.MODE_WORLD_READABLE+Context.MODE_WORLD_WRITEABLE);”。Android 有一套自己的安全模型，当应用程序(.apk)安装时系统就会分配给它一个 userid，当该应用要去访问其他资源例如文件的时候，就需要 userid 匹配。默认情况下，任何应用创建的文件，sharedpreferences 数据库都应该是私有的(位于/data/data/<package name>/files)，其他程序无法访问。

只有在创建时指定了 Context.MODE_WORLD_READABLE 或者 Context.MODE_WORLD_WRITEABLE，其他程序才能正确访问。

读取文件示例如下：

```
public void fileload()
{
    try {
```

```
            FileInputStream inStream= this.openFileInput("file.txt");
            ByteArrayOutputStream stream= new ByteArrayOutputStream();
            byte[] buffer= new byte[1024];
            int length= -1;
            while((length= inStream.read(buffer))!= -1) {
                stream.write(buffer,0,length);
            }
            stream.close();
            inStream.close();
            text.setText(stream.toString());
            Toast.makeText(MyActivity.this,"Loaded",Toast.LENGTH_LONG).show();
        } catch(FileNotFoundException e) {
            e.printStackTrace();
        }
        catch(IOException e){
            return ;
        }
    }
```

对于私有文件只能被创建该文件的应用访问,如果希望文件被其他应用读和写,可以在创建文件时指定 Context.MODE_WORLD_READABLE 和 Context.MODE_WORLD_WRITEABLE 权限。

Activity 还提供了 getCacheDir()和 getFilesDir()方法。getCacheDir()方法用于获取/data/data/<package name>/cache 目录;getFilesDir()方法用于获取/data/data/<package name>/files 目录。

SDCard 的存储空间比较大,在使用中应尽量把不重要的存入 SDCard。存入 SDCard 的方法如下:使用 Activity 的 openFileOutput()方法保存文件,文件是存放在手机空间上,一般手机的存储空间不是很大,存放些小文件还行,如果要存放像视频这样的大文件,是不可行的。对于像视频这样的大文件,可以把它存放在 SDCard 中。

SDCard 是干什么的?可以把它看作移动硬盘或 U 盘。在模拟器中使用 SDCard,需要先创建一张 SDCard 卡(当然不是真的 SDCard,只是镜像文件)。

创建 SDCard 可以在 Eclipse 创建模拟器时随同创建,也可以使用 DOS 命令进行创建,创建方法如下:

在 DOS 窗口中进入 Android SDK 安装路径的 tools 目录,输入以下命令创建一张容量为 2GB 的 SDCard,文件后缀可以随便取,建议使用.img:mksdcard 2048M,安装路径\sdcard.img。在程序中访问 SDCard,需要申请访问 SDCard 的权限。

在 AndroidManifest.xml 中加入访问 SDCard 的权限如下:

```
<!--在 SDCard 中创建与删除文件权限 -->
<uses-permission Android:name=
                    "Android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<!--向 SDCard 中写入数据权限 -->
```

```
<uses-permission Android:name=
                    "Android.permission.WRITE_EXTERNAL_STORAGE"/>
```

要向SDCard存放文件,程序必须先判断手机是否装有SDCard,并且可以进行读写。注意:访问SDCard必须在AndroidManifest.xml中加入访问SDCard的权限。下面是判断是否有SDCard的语句:

```
if(Environment.getExternalStorageState().equals(
            Environment.MEDIA_MOUNTED)){
    File sdCardDir= Environment.getExternalStorageDirectory();
    //获取SDCard目录
    File saveFile= new File(sdCardDir, "a.txt");
    FileOutputStream outStream= new FileOutputStream(saveFile);
    outStream.write("test".getBytes());
    outStream.close();
}
```

Environment.getExternalStorageState()方法用于获取SDCard的状态,如果手机装有SDCard,并且可以进行读写,那么方法返回的状态等于Environment.MEDIA_MOUNTED。

Environment.getExternalStorageDirectory()方法用于获取SDCard的目录,当然要获取SDCard的目录,也可以这样写:

```
File sdCardDir= new File("/sdcard");                //获取SDCard目录
File saveFile= new File(sdCardDir, "itcast.txt");
//上面两句代码可以合成一句
File saveFile= new File("/sdcard/a.txt");
FileOutputStream outStream= new FileOutputStream(saveFile);
outStream.write("test".getBytes());
outStream.close();
```

9.3 SQLite

SQLite是轻量级嵌入式数据库引擎,它支持SQL语言,并且只利用很少的内存就有很好的性能。此外它还是开源的,任何人都可以使用它。许多开源项目(Mozilla、PHP、Python)都使用了SQLite。SQLite由以下几个组件组成:SQL编译器、内核、后端和附件。SQLite通过利用虚拟机和虚拟数据库引擎(VDBE),使调试、修改和扩展SQLite的内核变得更加方便。如果想要开发Android应用程序,一定需要在Android上存储数据,使用SQLite数据库是一种非常好的选择。

SQLite的特点:面向资源有限的设备,没有服务器进程,所有数据存放在同一文件中,跨平台,可自由复制。

SQLite 内部结构如图 9.1 所示。

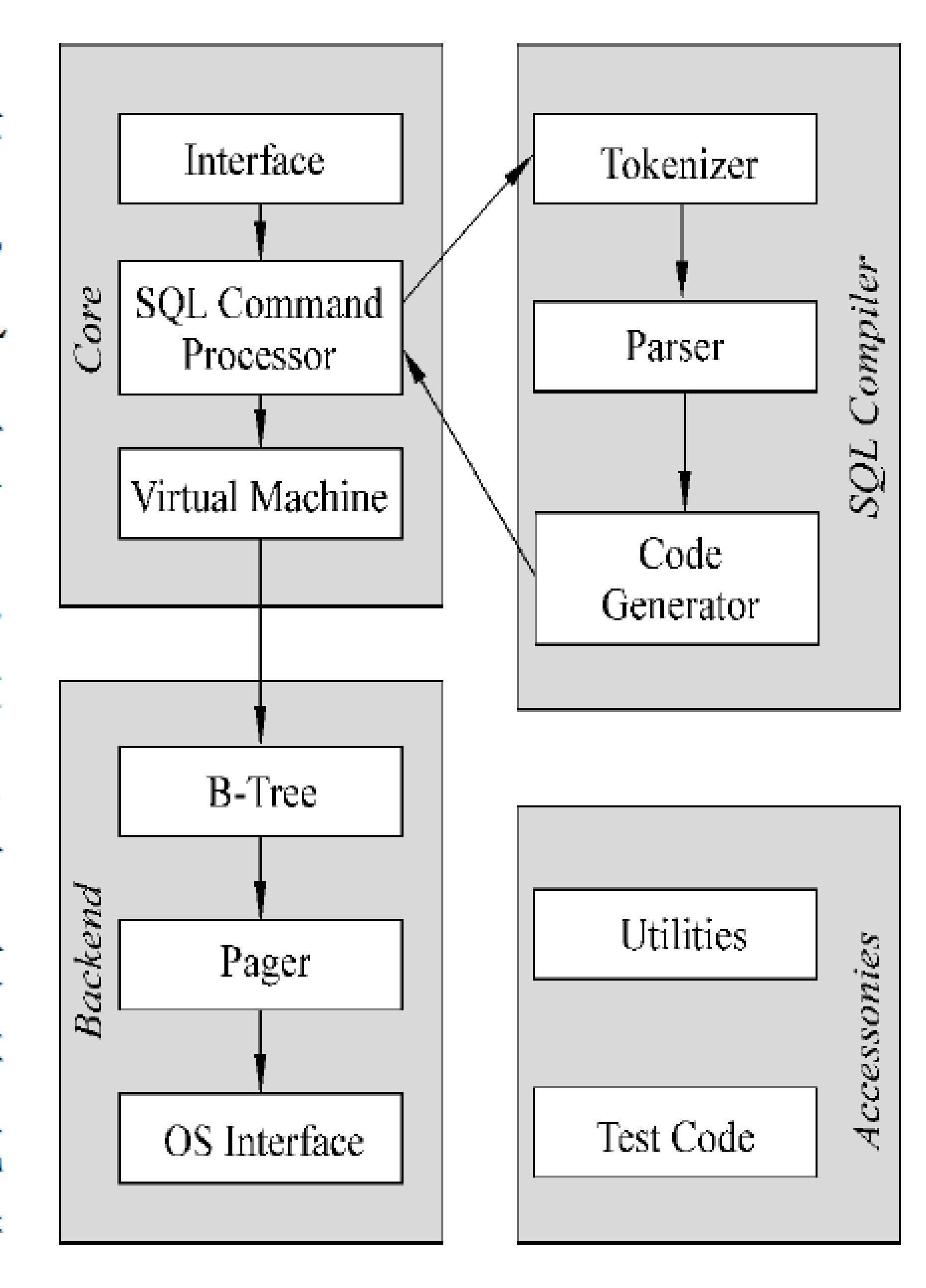


图 9.1　SQLite 内部结构

SQLite 基本上符合 SQL-92 标准，优点是高效，Android 运行时环境包含了完整的 SQLite。SQLite、SQLite3 支持 NULL、INTEGER、REAL(浮点数字)、TEXT(字符串文本)和 BLOB(二进制对象)数据类型。虽然它支持的类型只有 5 种，但实际上 SQLite3 也接受 varchar(n)、char(n)、decimal(p,s)等数据类型，只不过在运算或保存时会转成对应的 5 种数据类型。SQLite 最大的特点是"弱类型"，可以保存任何类型的数据到任何字段中，无论这列声明的数据类型是什么。例如，可以在 Integer 字段中存放字符串，或者在布尔型字段中存放浮点数，或者在字符型字段中存放日期型值。但有一种情况例外：定义为 INTEGER PRIMARY KEY 的字段只能存储 64 位整数，当向这种字段中保存除整数以外的数据时，将会产生错误。另外，SQLite 在解析 CREATE TABLE 语句时，会忽略 CREATE TABLE 语句中跟在字段名后面的数据类型信息。

SQLite 可以解析大部分标准 SQL 语句。

查询语句：

```
select * from 表名 where 条件子句 group by 分组字句
          having ... order by 排序子句
```

例如：

```
select * from person
select * from person order by id desc
select name from person group by name having count(*)>1
```

分页 SQL 与 MySQL 类似，下面 SQL 语句获取 5 条记录，跳过前面 3 条记录：

```
select * from Account limit 5 offset 3
```

或者：

```
select * from Account limit 3,5
```

插入语句：

```
insert into 表名(字段列表) values(值列表)
```

例如：

```
insert into person(name, age) values('传智',3)
```

更新语句：

```
update 表名 set 字段名=值 where 条件子句
```

例如：

```
update person set name='传智' where id=10
```

删除语句：

```
delete from 表名 where 条件子句
```

例如：

```
delete from person where id=10
```

由于JDBC会消耗太多的系统资源，所以JDBC对于手机这种内存受限设备来说并不合适。因此，Android提供了一些新的API来使用SQLite数据库。

数据库存储在data/＜应用程序主包＞/databases/下。

Android提供了SQLiteOpenHelper帮助创建一个数据库，只要继承SQLiteOpenHelper类，就可以轻松地创建数据库。SQLiteOpenHelper类根据开发应用程序的需要，封装了创建和更新数据库使用的逻辑。

9.3.1　SQLiteOpenHelper类

SQLiteOpenHelper类是用来实现创建、打开和升级数据库的最佳实践模式。通过实现SQLiteOpenHelper，可以隐藏那些用于决定一个数据库在被打开之前是否需要被创建或者升级的逻辑。

SQLiteOpenHelper的构造函数需要4个参数：上下文环境（例如一个Activity）、数据库名字、一个可选的游标工厂（通常是Null）和一个代表正在使用的数据库模型版本的整数。

abstract void onCreate(SQLiteDatabase db)：该函数第一次创建数据库的时候执行，实际上是在第一次得到SQLiteDatabase对象时才调用这个方法。它需要一个SQLiteDatabase对象作为参数，根据需要对这个对象填充表和初始化数据。

abstract void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)：数据库的版本发生变化时会被调用。它需要3个参数：一个SQLiteDatabase对象、一个旧的版本号和一个新的版本号。此方法在数据库版本每次发生变化时都会把用户手机上的数据库表删除，然后再重新创建，需要考虑用户存放于数据库中的数据是否丢失。

调用SQLiteOpenHelper的getWritableDatabase()或者getReadableDatabase()方法获取用于操作数据库的SQLiteDatabase实例。

synchronized SQLiteDatabase getReadableDatabase()：先以读写方式打开数据库，如果数据库的磁盘空间满了，就会打开失败，当打开失败后会继续尝试以只读方式打开数据库：

```
return(db==null)?false : true;
```

synchronized SQLiteDatabase getWritableDatabase()：以读写方式打开数据库，一旦数据库的磁盘空间满了，数据库就只能读而不能写，倘若使用的是 getWritableDatabase() 方法就会出错。

获得 SQLiteDatabase 的代码如下：

```
db= (new DatabaseHelper(getContext())).getWritableDatabase();
```

由于磁盘空间或者权限问题，对 getWritableDatabase()的调用可能会失败，所以需要提供一个对 getReadableDatabase 方法的回退(fallback)：

```
dbHelper= new myDbHelper(context, DATABASE_NAME, null, DATABASE_VERSION);
SQLiteDatabase db;
try {
        db= dbHelper.getWritableDatabase();
}
catch(SQLiteException ex){
        db= dbHelper.getReadableDatabase();
}
```

下面举例说明如何继承 SQLiteOpenHelper 创建数据库：

```
public class DatabaseHelper extends SQLiteOpenHelper {
      DatabaseHelper(Context context, String name,
                          CursorFactory cursorFactory, int version)
      {
            super(context, name, cursorFactory, version);
      }
      @Override
      public void onCreate(SQLiteDatabase db) {
            //TODO 创建数据库后,对数据库的操作 }
            @Override
            public void onUpgrade(SQLiteDatabase db, int oldVersion,
                                        int newVersion) {//TODO 更改数据库版本的操作 }
            @Override
            public void onOpen(SQLiteDatabase db) {
                  super.onOpen(db);              //TODO 每次成功打开数据库后首先被执行
      }
}
```

9.3.2 SQLiteDatabase 类

SQLiteDatabase 类封装了一些操作数据库的 API，使用该类可以对数据进行添加(Create)、查询(Retrieve)、更新(Update)和删除(Delete)操作(这些操作简称为 CRUD)。

```
Void execSQL(String sql)
void execSQL(String sql, Object[] bindArgs)
```

```
Cursor rawQuery(String sql, String[] selectionArgs)
Cursor rawQueryWithFactory(SQLiteDatabase.CursorFactory cursorFactory,
                           String sql, String[] selectionArgs, String editTable)
```

当完成了对数据库的操作(例如 Activity 已经关闭),需要调用 SQLiteDatabase 的 Close()方法来释放数据库连接。

创建表和索引需要调用 SQLiteDatabase 的 execSQL()方法来执行 DDL 语句。如果没有异常,这个方法没有返回值。

例如:

```
db.execSQL("CREATE TABLE mytable(_id INTEGER PRIMARY KEY AUTOINCREMENT,
                 title TEXT, value REAL);");
```

删除表和索引需要使用 execSQL()方法调用 DROP INDEX 和 DROP TABLE 语句。

给表添加数据有两种方法可以实现:

(1) 可以使用 execSQL()方法执行 INSERT、UPDATE、DELETE 等语句来更新表的数据。execSQL()方法适用于所有不返回结果的 SQL 语句。

例如:

```
db.execSQL("INSERT INTO widgets(name, inventory)"+
      "VALUES('Sprocket', 5)");
db.execSQL("insert into widgets(name, inventory) values(?,?)",
      new Object[]{"Sprocket", 5});
```

(2) 另一种方法是使用 SQLiteDatabase 对象的 insert()、update()、delete()方法。这些方法把 SQL 语句的一部分作为参数。

示例如下:

```
ContentValues cv= new ContentValues();
cv.put(Constants.TITLE, "example title");
cv.put(Constants.VALUE, SensorManager.GRAVITY_DEATH_STAR_I);
db.insert("mytable", getNullColumnHack(), cv);
```

update()方法有 4 个参数,分别是表名、表示列名和值的 ContentValues 对象、可选的 WHERE 条件和可选的填充 WHERE 语句的字符串,这些字符串会替换 WHERE 条件中的"?"标记。

例如:

```
String[] parms= new String[] {"this is a string"};
db.update("widgets", replacements, "name=?", parms);
```

delete()方法的使用和 update()类似,使用表名、可选的 WHERE 条件和相应的填充 WHERE 条件的字符串。

查询数据库有两种方法使用 SELECT 从 SQLite 数据库检索数据:

(1) 使用 rawQuery()直接调用 SELECT 语句。这是最简单的解决方法,通过这个方法可以调用 SQL SELECT 语句。

例如:

```
Cursor c= db.rawQuery("SELECT name FROM sqlite_master WHERE
             type= 'table' AND name= 'mytable'", null);
```

(2) 使用 query()方法构建一个查询。当需要查询的列在程序编译的时候不能确定,这时使用 query()方法会方便很多。

Regular Queries query()方法用 SELECT 语句段构建查询。SELECT 语句内容作为 query()方法的参数,例如要查询的表名、要获取的字段名、WHERE 条件、包含可选的位置参数、去替代 WHERE 条件中位置参数的值、GROUP BY 条件、HAVING 条件。除了表名,其他参数可以是 null。所以,以前的代码段可以写成:

```
String[] columns= {"ID", "inventory"};
String[] parms= {"snicklefritz"};
Cursor result= db.query("widgets", columns, "name= ?",parms,
                    null, null, null);
```

9.3.3 Cursor 接口

Cursor 是一个游标接口,相当于指向底层数据中结果集的指针,游标是在数据库查询的结果集中对位置(行)的控制方式。有以下方法。

- abstract boolean moveToFirst():把游标移动到查询结果集的第一行。
- abstract boolean moveToNext():把游标移动到下一行。
- abstract boolean moveToPrevious():把游标移动到前一行。
- abstract int getCount():返回结果集的行数。
- abstract boolean moveToPosition(int position):移动到指定行的游标。
- abstract int getPosition():返回当前的游标的位置。
- abstract boolean isFirst():判断是否是第一条记录。
- abstract boolean isLast():判断是否是最后一条记录。
- Cursor query (String table, String [] columns, String selection, String [] selectionArgs, String groupBy, String having, String orderBy, String limit)。
- Cursor query (String table, String [] columns, String selection, String [] selectionArgs, String groupBy, String having, String orderBy)。
- Cursor query(boolean distinct, String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)。
- distinct:一个可选的布尔值,用来说明返回的值是否只包含唯一的值。
- table:要查询的表的名称。
- columns:一个投影,它作为一个字符数组,列出了只包含在结果集中的类。

- selection：一个 where 子句定义了要返回的行，可以在其中包含"?"通配符，它将会被存储在选择参数中的值替换。
- selectionArgs：一个选择参数字符串的数组，它将会替换 where 子句中的"?"。
- groupBy：一个 group by 子句，用来定义返回的行的分组方式。
- having：一个 having 过滤器，如果包含了一个 group by 子句，则这个过滤器会定义要包含哪些行的分组。
- orderBy：一个字符串，用来描述要返回的行的顺序。
- limit：一个可选的字符串，用来定义对返回的行数的限制。

下面举例说明如何遍历 mytable 表：

```
Cursor result= db.rawQuery("SELECT ID, name, inventory FROM mytable");
result.moveToFirst();
while(!result.isAfterLast()) {
    int id= result.getInt(0);
    String name= result.getString(1);
    int inventory= result.getInt(2);
    //do something useful with these
    result.moveToNext();
}
result.close();
```

从 Cursor 中提取结果：

```
int GOLD_HOARDED_COLUMN= 2;
Cursor myGold= myDatabase.query("GoldHoards", null, null, null,
                    null, null, null);
float totalHoard= 0f;
//Make sure there is at least one row.
if(myGold.moveToFirst()) {
    //Iterate over each cursor.
    do {
        float hoard= myGold.getFloat(GOLD_HOARDED_COLUMN);
        totalHoard+ = hoard;
    } while(myGold.moveToNext());
}
```

9.3.4 标准数据库 adapter 类的实现代码

```
import Android.content.Context;
import Android.database.*;
import Android.database.sqlite.*;
import Android.database.sqlite.SQLiteDatabase.CursorFactory;
import Android.util.Log;
public class MyDBAdapter {
```

```
    private static final String DATABASE_NAME= "myDatabase.db";
    private static final String DATABASE_TABLE= "mainTable";
    private static final int DATABASE_VERSION= 1;
    //Where子句中使用的索引(键)列的名称
    public static final String KEY_ID= "_id";
    //数据库每个列的名称和索引
  public static final String KEY_NAME= "name";
    public static final int NAME_COLUMN= 1;
    //TODO: Create public field for each column in your table.
    //SQL Statement to create a new database.
    private static final String DATABASE_CREATE= "create table "+
      DATABASE_TABLE+ "("+ KEY_ID+
      " integer primary key autoincrement, "+
      KEY_NAME+ " text not null);";
    //Variable to hold the database instance
    private SQLiteDatabase db;
    //Context of the application using the database.
    private final Context context;
    //Database open/upgrade helper
    private myDbHelper dbHelper;
    public MyDBAdapter(Context _context) {
      context= _context;
      dbHelper= new myDbHelper(context, DATABASE_NAME,
                                null, DATABASE_VERSION);
    }
    public MyDBAdapter open() throws SQLException {
      db= dbHelper.getWritableDatabase();
      return this;
    }
    public void close() {
        db.close();
    }
    public int insertEntry(MyObject _myObject) {
      //TODO: Create a new ContentValues to represent my row
      //and insert it into the database.
      return index;
    }
    public boolean removeEntry(long _rowIndex) {
      return db.delete(DATABASE_TABLE, KEY_ID+ "= "+ _rowIndex, null) > 0;
    }
    public Cursor getAllEntries() {
      return db.query(DATABASE_TABLE, new String[] {KEY_ID, KEY_NAME},
                          null, null, null, null, null);
    }
    public MyObject getEntry(long _rowIndex) {
      //TODO: Return a cursor to a row from the database and
```

```
        //use the values to populate an instance of MyObject
        return objectInstance;
    }
    public boolean updateEntry(long _rowIndex, MyObject _myObject) {
        //TODO: Create a new ContentValues based on the new object
        //and use it to update a row in the database.
        return true;
    }
    private static class myDbHelper extends SQLiteOpenHelper {
        public myDbHelper(Context context, String name,
                          CursorFactory factory, int version) {
            super(context, name, factory, version);
        }
        //Called when no database exists in disk and the helper class needs
        //to create a new one.
        @Override
        public void onCreate(SQLiteDatabase _db) {
            _db.execSQL(DATABASE_CREATE);
        }
        //Called when there is a database version mismatch meaning that the version
        //of the database on disk needs to be upgraded to the current version.
        @Override
        public void onUpgrade(SQLiteDatabase _db, int _oldVersion,
                              int _newVersion) {
            //Log the version upgrade.
            Log.w("TaskDBAdapter", "Upgrading from version "+
                            _oldVersion+ " to "+
                            _newVersion+ ",
                             which will destroy all old data");
            //Upgrade the existing database to conform to the new version. Multiple
            //previous versions can be handled by comparing _oldVersion and
            //_newVersion values.
            //The simplest case is to drop the old table and create a new one.
            _db.execSQL("DROP TABLE IF EXISTS "+ DATABASE_TABLE);
            //Create a new one.
            onCreate(_db);
        }
    }
}
```

9.3.5　注意事项

(1) 文件(例如位图、或者音频文件)通常是不存储在数据库文件中的。

(2) 强烈建议所有表都应该包含一个自动增加的键域,从而为每一行提供唯一的

索引。

(3) 第一次调用 getWritableDatabase() 或 getReadableDatabase() 方法后，SQLiteOpenHelper 会缓存当前的 SQLiteDatabase 实例，SQLiteDatabase 实例正常情况下会维持数据库的打开状态，所以当不再需要 SQLiteDatabase 实例时，请及时调用 close() 方法释放资源。一旦 SQLiteDatabase 实例被缓存，多次调用 getWritableDatabase() 或 getReadableDatabase() 方法得到的都是同一实例。

(4) 使用 SQLiteDatabase 的 beginTransaction() 方法可以开启一个事务，程序执行到 endTransaction() 方法时会检查事务的标志是否为成功，如果为成功则提交事务，否则回滚事务。当应用需要提交事务，必须在程序执行到 endTransaction() 方法之前使用 setTransactionSuccessful() 方法设置事务的标志为成功，如果不调用 setTransactionSuccessful() 方法，默认会回滚事务。使用例子如下：

```
SQLiteDatabase db= ....;
db.beginTransaction();                  //开始事务
try {
    db.execSQL("insert into person(name, age) values(?,?)",
            new Object[]{"传智播客", 4});
    db.execSQL("update person set name= ?where personid= ?",
            new Object[]{"传智", 1});
    db.setTransactionSuccessful();      //调用此方法会在执行到 endTransaction()时
                                        //提交当前事务,如果不调用此方法会回滚事务
} finally {
    db.endTransaction();                //由事务的标志决定是提交事务,还是回滚事务
}
db.close();
```

上面两条 SQL 语句在同一个事务中执行。

(5) 使用 Android 模拟器有两种可供选择的方法来管理数据库。

首先，模拟器绑定了 sqlite3 控制台程序，可以使用 adb shell 命令来调用它。只要进入了模拟器的 shell，在数据库的路径执行 sqlite3 命令就可以了。

数据库文件一般存放在/data/data/your. app. package/databases/your-db-name 中，如果喜欢使用更友好的工具，可以把数据库复制到开发机上，使用 SQLite-aware 客户端来操作它。这样的话，在一个数据库的拷贝上操作，如果想要修改能反映到设备上，需要把数据库备份回去。

把数据库从设备上复制出来，可以使用 adb pull 命令(或者在 IDE 上做相应操作)。

存储一个修改过的数据库到设备上，使用 adb push 命令。一个最方便的 SQLite 客户端是 FireFox SQLite Manager 扩展，它可以跨所有平台使用。

9.4 使用 ContentProvider 存储数据

数据在 Android 中是私有的，这些数据包括文件数据和数据库数据以及一些其他类型的数据。解决两个程序之间交换数据这个问题主要靠 ContentProvider。一个

ContentProvider 类实现了一组标准的方法接口，从而能够让其他的应用保存或读取此 Content Provider 的各种数据类型。也就是说，一个程序可以通过实现一个 Content Provider 的抽象接口将自己的数据暴露出去。外界根本看不到，也不用看到这个应用暴露的数据在应用当中是如何存储的，是用数据库存储，还是用文件存储，还是通过网上获得，这些都不重要，重要的是外界可以通过这一套标准及统一的接口和程序里的数据打交道，可以读取程序的数据，也可以删除程序的数据，当然，中间也会涉及一些权限的问题。

ContentProvider 就像一个"数据库"。外界获取其提供的数据，也就应该与从数据库中获取数据的操作基本一样，只不过是采用 URI 来表示外界需要访问的"数据库"。

Content Provider 提供了一种多应用间数据共享的方式，例如，联系人信息可以被多个应用程序访问。

Content Provider 是个实现了一组用于提供其他应用程序存取数据的标准方法的类。应用程序可以在 Content Provider 中执行如下操作：查询数据、修改数据、添加数据和删除数据。

标准的 Content Provider：Android 提供了一些已经在系统中实现的标准 ContentProvider，例如联系人信息、图片库等，可以用这些 Content Provider 来访问设备上存储的联系人信息、图片等。

9.4.1 使用 ContentProvider 共享数据

当应用继承 ContentProvider 类，并重写该类用于提供数据和存储数据的方法，就可以向其他应用共享其数据。虽然使用其他方法也可以对外共享数据，但数据访问方式会因数据存储的方式而不同，例如，采用文件方式对外共享数据，需要进行文件操作读写数据；采用 sharedpreferences 共享数据，需要使用 sharedpreferences API 读写数据。而使用 ContentProvider 共享数据的好处是统一了数据访问方式。当应用需要通过 ContentProvider 对外共享数据时，需要以下步骤：

(1) 继承 ContentProvider 并重写下列方法：

```
public class PersonContentProvider extends ContentProvider{
    public boolean onCreate();
    public Uri insert(Uri uri, ContentValues values);
    public int delete(Uri uri, String selection, String[] selectionArgs);
    public int update(Uri uri, ContentValues values, String selection,
                        String[] selectionArgs);
    public Cursor query(Uri uri, String[] projection, String selection,
                        String[] selectionArgs, String sortOrder);
    public String getType(Uri uri);
}
```

(2) 在 AndroidManifest. xml 中使用<provider>对该 ContentProvider 进行配置，为了能让其他应用找到该 ContentProvider，ContentProvider 采用了 authorities(主机名/域名)对它进行唯一标识，可以把 ContentProvider 看做一个网站(网站也是提供数据者)，

authorities 就是它的域名：

```
<manifest ...>
<application Android:icon="@drawable/icon"
                    Android:label="@string/app_name">
<provider Android:name=".PersonContentProvider"
                     Android:authorities="com.example.transportationprovider"/>
</application>
</manifest>
```

注意：一旦应用继承了 ContentProvider 类，后面就会把这个应用称为 ContentProvider (内容提供者)。

ContentProvider 类主要方法的作用如下。

- public boolean onCreate()：该方法在 ContentProvider 创建后就会被调用，Android 在系统启动时就会创建 ContentProvider。
- public Uri insert(Uri uri, ContentValues values)：该方法用于供外部应用向 ContentProvider 添加数据。
- public int delete(Uri uri, String selection, String[] selectionArgs)：该方法用于供外部应用从 ContentProvider 删除数据。
- public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)：该方法用于供外部应用更新 ContentProvider 中的数据。
- public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)：该方法用于供外部应用从 ContentProvider 中获取数据。
- public String getType(Uri uri)：该方法用于返回当前 Uri 所代表数据的 MIME 类型。如果操作的数据属于集合类型，那么 MIME 类型字符串应该以 vnd.Android.cursor.dir/开头，例如，要得到所有 person 记录的 Uri 为 content://cn.togogo.provider.personprovider/person，那么返回的 MIME 类型字符串应该为"vnd.Android.cursor.dir/person"。如果要操作的数据属于单一数据，那么 MIME 类型字符串应该以 vnd.Android.cursor.item/开头，例如，得到 id 为 10 的 person 记录，Uri 为 content://cn.togogo.provider.personprovider/person/10，那么返回的 MIME 类型字符串应该为"vnd.Android.cursor.item/person"。

下面分功能介绍。

1. 创建 Content Provider

要创建自己的 Content Provider，需要遵循以下几步。

(1) 创建一个继承了 ContentProvider 父类的类。

(2) 定义一个名为 CONTENT_URI，并且是 public static final 的 Uri 类型的类变量，必须为其指定一个唯一的字符串值，最好的方案是以类的全名称，例如 public static。

```
final Uri CONTENT_URI=
     Uri.parse("content://com.google.Android.MyContentProvider");
```

(3) 创建数据存储系统。大多数 Content Provider 使用 Android 文件系统或 SQLite 数据库来保持数据,但是也可以以任何想要的方式来存储。

(4) 定义要返回给客户端的数据列名。如果正在使用 Android 数据库,则数据列的使用方式就和以往所熟悉的其他数据库一样。但是必须为其定义一个叫_id 的列,用来表示每条记录的唯一性。

(5) 如果要存储字节型数据,比如位图文件等,那么保存该数据的数据列其实是一个表示实际保存文件的 URI 字符串,客户端通过它来读取对应的文件数据,处理这种数据类型的 Content Provider 需要实现一个名为_data 的字段,_data 字段列出了该文件在 Android 文件系统上的精确路径。这个字段不仅供客户端使用,而且也可以供 ContentResolver 使用。客户端可以调用 ContentResolver. openOutputStream()方法来处理该 URI 指向的文件资源,如果是 ContentResolver 本身的话,由于其持有的权限比客户端要高,所以它能直接访问这些数据文件。

(6) 声明 public static String 型的变量,用于指定要从游标处返回的数据列。

(7) 查询返回一个 Cursor 类型的对象。所有执行写操作的方法如 insert()、update()以及 delete()都将被监听。可以通过使用 ContentResover(). notifyChange()方法来通知监听器关于数据更新的信息。

(8) 在 AndroidMenifest. xml 中使用标签来设置 Content Provider。

2. 查询记录

在 Content Provider 中使用的查询字符串有别于标准的 SQL 查询。很多诸如 select、add、delete、modify 等操作都使用一种特殊的 URI 来进行。这种 URI 由 3 个部分组成:“content://”、代表数据的路径和一个可选的标识数据的 ID。

以下是一些示例 URI。

- content://media/internal/images: 这个 URI 将返回设备上存储的所有图片。
- content://contacts/people/: 这个 URI 将返回设备上所有的联系人信息。
- content://contacts/people/45: 这个 URI 将返回单个结果(联系人信息中 ID 为 45 的联系人记录)。

尽管这种查询字符串格式很常见,但是它看起来还是有点令人迷惑。为此,Android 提供一系列的帮助类(在 Android. provider 包下),里面包含了很多以类变量形式给出的查询字符串,这种方式更容易让我们理解一点,参见下面示例:

```
MediaStore.Images.Media.INTERNAL_CONTENT_URI Contacts.People.CONTENT_URI
```

因此,如上面 content://contacts/people/45 这个 URI 就可以写成如下形式:

```
Uri person= ContentUris.withAppendedId(People.CONTENT_URI, 45);
```

然后执行数据查询:

```
Cursor cur= managedQuery(person, null, null, null);
```

这个查询返回一个包含所有数据字段的游标,可以通过迭代这个游标来获取所有的数据。下面说明如何依次读取联系人信息表中的指定数据列 name 和 number:

```
public class ContentProviderDemo extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        displayRecords();
    }
    private void displayRecords() {
        //该数组中包含了所有要返回的字段
        String columns[]= new String[] {People.NAME, People.NUMBER};
        Uri mContacts= People.CONTENT_URI;
        Cursor cur= managedQuery(
                        mContacts,
                        columns,            //要返回的数据字段
                        null,               //WHERE子句
                        null,               //WHERE子句的参数
                        null                //Order-by子句
        );
        if(cur.moveToFirst()) {
            String name= null;
            String phoneNo= null;
            do {
                //获取字段的值
                name= cur.getString(cur.getColumnIndex(People.NAME));
                phoneNo= cur.getString(cur.getColumnIndex(People.NUMBER));
                Toast.makeText(this, name+ " "+ phoneNo,
                                    Toast.LENGTH_LONG).show();
            } while(cur.moveToNext());
        }
    }
}
```

3. 修改记录

可以使用 ContentResolver.update()方法修改数据。下面来写一个修改数据的方法：

```
private void updateRecord(int recNo, String name) {
    Uri uri= ContentUris.withAppendedId(People.CONTENT_URI, recNo);
    ContentValues values= new ContentValues();
    values.put(People.NAME, name);
    getContentResolver().update(uri, values, null, null);
}
```

现在可以调用上面的方法来更新指定记录：

```
updateRecord(10, "XYZ");                    //更改第10条记录的name字段值为"XYZ"
```

4. 添加记录

要增加记录,可以调用 ContentResolver.insert()方法,该方法接受一个要增加的记录的目标 URI,以及一个包含了新记录值的 Map 对象,调用后的返回值是新记录的 URI,包含记录号。

上面的例子中都是基于联系人信息簿这个标准的 Content Provider,现在创建一个 insertRecord() 方法对联系人信息簿进行数据的添加:

```
private void insertRecords(String name, String phoneNo) {
    ContentValues values= new ContentValues();
    values.put(People.NAME, name);
    Uri uri= getContentResolver().insert(People.CONTENT_URI, values);
    Log.d("Android", uri.toString());
    Uri numberUri= Uri.withAppendedPath(uri,
            People.Phones.CONTENT_DIRECTORY);
    values.clear();
    values.put(Contacts.Phones.TYPE, People.Phones.TYPE_MOBILE);
    values.put(People.NUMBER, phoneNo);
    getContentResolver().insert(numberUri, values);
}
```

这样就可以调用 insertRecords(name,phoneNo)方法向联系人信息簿中添加联系人姓名和电话号码了。

5. 删除记录

Content Provider 中的 getContextResolver.delete()方法可以用来删除记录。

下面的记录用来删除设备上所有的联系人信息:

```
private void deleteRecords() {
      Uri uri= People.CONTENT_URI;
      getContentResolver().delete(uri, null, null);
}
```

也可以指定 WHERE 条件语句来删除特定的记录:

```
getContentResolver().delete(uri, "NAME= "+ "'XYZ XYZ'", null);
```

这将会删除 name 为'XYZ XYZ'的记录。

6. 自定义数据类型

如果要处理的数据类型是一种比较新的类型,就必须先定义一个新的 MIME 类型,以供 ContentProvider.geType(url)来返回。

MIME 类型有两种形式:一种是为指定的单个记录的;一种是为多条记录的。这里给出常用的格式:

(1) vnd.Android.cursor.item/vnd.yourcompanyname.contenttype(单个记录的 MIME

类型)。例如,一个请求列车信息的 URI 为 content://com. example. transportationprovider/trains/122,可能就会返回 typevnd. Android. cursor. item/vnd. example. rail 这样一个 MIME 类型。

(2) vnd. Android. cursor. dir/vnd. yourcompanyname. contenttype(多个记录的 MIME 类型)。例如,一个请求所有列车信息的 URI 为 content://com. example. transportationprovider/trains,可能就会返回 vnd. Android. cursor. dir/vnd. example. rail 这样一个 MIME 类型。

下列代码将创建一个 Content Provider,它仅仅存储用户名称并显示所有的用户名称(使用 SQLLite 数据库存储这些数据):

```
public class MyUsers {
    public static final String AUTHORITY= "com.wissen.MyContentProvider";
    //BaseColumn类中已经包含了 _id字段
    public static final class User implements BaseColumns {
        public static final Uri CONTENT_URI=
                Uri.parse("content://com.wissen.MyContentProvider");
        //表数据列
        public static final String USER_NAME= "USER_NAME";
    }
}
```

上面的类中定义了 Content Provider 的 CONTENT_URI,以及数据列。下面将基于上面的类来定义实际的 Content Provider 类:

```
public class MyContentProvider extends ContentProvider {
    private SQLiteDatabase sqlDB;
    private DatabaseHelper dbHelper;
    private static final String DATABASE_NAME= "Users.db";
    private static final int DATABASE_VERSION= 1;
    private static final String TABLE_NAME= "User";
    private static final String TAG= "MyContentProvider";
    private static class DatabaseHelper extends SQLiteOpenHelper {
        DatabaseHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }
        @Override
        public void onCreate(SQLiteDatabase db) {
            //创建用于存储数据的表
            db.execSQL("Create table "+ TABLE_NAME+
             "(_id INTEGER PRIMARY KEY AUTOINCREMENT, USER_NAME TEXT);");
        }
        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion,
                    int newVersion){
```

```
            db.execSQL("DROP TABLE IF EXISTS "+ TABLE_NAME);
            onCreate(db);
        }
    }
    @Override
    public int delete(Uri uri, String s, String[] as) {
        return 0;
    }
    @Override
    public String getType(Uri uri) {
        return null;
    }
    @Override
    public Uri insert(Uri uri, ContentValues contentvalues) {
        sqlDB= dbHelper.getWritableDatabase();
        long rowId= sqlDB.insert(TABLE_NAME, "", contentvalues);
        if(rowId > 0) {
            Uri rowUri= ContentUris.appendId(
                MyUsers.User.CONTENT_URI.buildUpon(), rowId).build();
            getContext().getContentResolver().notifyChange(rowUri, null);
            return rowUri;
        }throw new SQLException("Failed to insert row into "+ uri);
    }
    @Override
    public boolean onCreate() {
        dbHelper= new DatabaseHelper(getContext());
        return(dbHelper= = null) ?false : true;
    }
    @Override
    public Cursor query(Uri uri, String[] projection, String selection,
            String[] selectionArgs, String sortOrder) {
        SQLiteQueryBuilder qb= new SQLiteQueryBuilder();
        SQLiteDatabase db= dbHelper.getReadableDatabase();
        qb.setTables(TABLE_NAME);
        Cursor c= qb.query(db, projection, selection, null,
                                    null, null, sortOrder);
        c.setNotificationUri(getContext().getContentResolver(), uri);
        return c;
    }
    @Override
    public int update(Uri uri, ContentValues contentvalues, String s,
                    String[] as) {
        return 0;
    }
```

```
}
```

一个名为 MyContentProvider 的 Content Provider 创建完成了，它用于从 SQLite 数据库中添加和读取记录。

```
public class MyContentDemo extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        insertRecord("MyUser");
        displayRecords();
    }
    private void insertRecord(String userName) {
        ContentValues values= new ContentValues();
        values.put(MyUsers.User.USER_NAME, userName);
        getContentResolver().insert(MyUsers.User.CONTENT_URI, values);
    }
    private void displayRecords() {
        String columns[]= new String[] {MyUsers.User._ID,
                                        MyUsers.User.USER_NAME };
        Uri myUri= MyUsers.User.CONTENT_URI;
        Cursor cur= managedQuery(myUri, columns,null, null, null);
        if(cur.moveToFirst()) {
            String id= null;
            String userName= null;
            do {
                id= cur.getString(cur.getColumnIndex(MyUsers.User._ID));
                userName= cur.getString(
                                cur.getColumnIndex(MyUsers.User.USER_NAME));
                Toast.makeText(this, id+ " "+ userName,
                          Toast.LENGTH_LONG).show();
            } while(cur.moveToNext());
        }
    }
}
```

上面的类先向数据库中添加一条用户数据，然后显示数据库中所有的用户数据。

9.4.2 Uri 介绍

Uri 代表了要操作的数据，Uri 主要包含两部分信息：

(1) 需要操作的 ContentProvider。

(2) 对 ContentProvider 中的什么数据进行操作。

一个 Uri 由几部分组成，如图 9.2 所示。

content://com.example.transportationprovider/trains/122
A B C D

图 9.2 Uri 组成

下面按照图 9.2 中标示的 A、B、C、D 来分块说明。

- A：标准前缀，用来说明一个 Content Provider 控制这些数据，无法改变的；"content://"。
- B：URI 的标识，它定义了是哪个 Content Provider 提供这些数据。对于第三方应用程序，为了保证 URI 标识的唯一性，它必须是一个完整的、小写的类名。这个标识在元素的 authorities 属性中说明：一般是定义该 ContentProvider 的包.类的名称；"content://hx.Android.text.myprovider"。
- C：路径，通俗的讲就是要操作的数据库中表的名字，或者也可以自己定义，记得在使用的时候保持一致就可以了；"content://hx.Android.text.myprovider/tablename"。
- D：如果 URI 中包含表示需要获取的记录的 ID；则就返回该 id 对应的数据，如果没有 ID，就表示返回全部；"content://hx.Android.text.myprovider/tablename/#" #表示数据 id。

将字符串转成 Uri，Uri uri=Uri.parse("content://cn.togogo.provider.personprovider/person")。

使用 ContentResolver 操作 ContentProvider 中的数据：

当外部应用需要对 ContentProvider 中的数据进行添加、删除、修改和查询操作时，可以使用 ContentResolver 类来完成。要获取 ContentResolver 对象，可以使用 Activity 提供的 getContentResolver()方法。ContentResolver 类提供了与 ContentProvider 类相同签名的 4 个方法。

(1) public Uri insert(Uri uri, ContentValues values)：该方法用于向 ContentProvider 添加数据。

(2) public int delete(Uri uri, String selection, String[] selectionArgs)：该方法用于从 ContentProvider 删除数据。

(3) public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)：该方法用于更新 ContentProvider 中的数据。

(4) public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)：该方法用于从 ContentProvider 中获取数据。

这些方法的第一个参数为 Uri，代表要操作的是哪个 ContentProvider 和对其中的什么数据进行操作，假设给定的是 Uri.parse("content://cn.togogo.provider.personprovider/person/10")，那么将会对主机名为 cn.togogo.provider.personprovider 的 ContentProvider 进行操作，操作的数据为 person 表中 id 为 10 的记录。

使用 ContentResolver 对 ContentProvider 中的数据进行添加、删除、修改和查询操作：

```
ContentResolver resolver= getContentResolver();
Uri uri= Uri.parse("content://cn.togogo.provider.personprovider/person");
//添加一条记录
ContentValues values= new ContentValues();
values.put("name", "togogo");
values.put("age", 25);
resolver.insert(uri, values);
//获取 person 表中所有记录
Cursor cursor= resolver.query(uri, null, null, null, "personid desc");
while(cursor.moveToNext()){
    Log.i("ContentTest", "personid= "+ cursor.getInt(0)+ ",
            name= "+ cursor.getString(1));
}
//把 id为 1的记录的 name字段值更新为 liming
ContentValues updateValues= new ContentValues();
updateValues.put("name", "liming");
Uri updateIdUri= ContentUris.withAppendedId(uri, 2);
resolver.update(updateIdUri, updateValues, null, null);
//删除 id为 2的记录
Uri deleteIdUri= ContentUris.withAppendedId(uri, 2);
resolver.delete(deleteIdUri, null, null);
```

1. UriMatcher 类使用介绍(public class UriMatcher)

因为 Uri 代表了要操作的数据,所以经常需要解析 Uri,并从 Uri 中获取数据。Android 系统提供了两个用于操作 Uri 的工具类:UriMatcher 和 ContentUris。

使用方法如下:

```
//常量 UriMatcher.NO_MATCH表示不匹配任何路径的返回码
UriMatcher sMatcher= new UriMatcher(UriMatcher.NO_MATCH);
//如果 match()方法匹配 content://cn.togogo.provider.personprovider/person
//路径,返回匹配码为 1
sMatcher.addURI("cn.togogo.provider.personprovider", "person", 1);
//添加需要匹配 uri,如果匹配就会返回匹配码
//如果 match()方法与 content://cn.togogo.provider.personprovider/person/230
//路径匹配,返回匹配码为 2
sMatcher.addURI("cn.togogo.provider.personprovider", "person/# ", 2);
//# 号为通配符
switch(sMatcher.match(Uri.parse(
            "content://cn.togogo.provider.personprovider/person/10"))) {
    case 1
        break;
    case 2
        break;
    default://不匹配
```




```
            break;
    }
```

注册完需要匹配的 Uri 后，就可以使用 sMatcher.match(uri)方法对输入的 Uri 进行匹配，如果匹配就返回匹配码，匹配码是调用 addURI()方法传入的第三个参数。假设匹配 content://cn.togogo.provider.personprovider/person 路径，返回的匹配码为 1。

2. ContentUris 类使用介绍

ContentUris 类用于获取 Uri 路径后面的 ID 部分，它有两个比较实用的方法：withAppendedId(uri,id)和 parseId(uri)。

(1) withAppendedId(uri, id)方法用于为路径加上 ID 部分：

```
Uri uri=Uri.parse("content://cn.togogo.provider.personprovider/person")
Uri resultUri=ContentUris.withAppendedId(uri, 10);
//生成后的 Uri 为: content://cn.togogo.provider.personprovider/person/10
```

(2) parseId(uri)方法用于从路径中获取 ID 部分：

```
Uri uri=
    Uri.parse("content://cn.togogo.provider.personprovider/person/10")
long personid=ContentUris.parseId(uri);            //获取的结果为 10
```

3. SQLiteQueryBuilder 类使用介绍

```
public static HashMap<String, String> userProjectionMap;
static {
    userProjectionMap=new HashMap<String, String>();
    userProjectionMap.put(UserData._ID, UserData._ID);
    userProjectionMap.put(UserData.USER_NAME,UserData.USER_NAME);
}
SQLiteQueryBuilder qb=new SQLiteQueryBuilder();
//设置查询哪张表
qb.setTables(UserData.TABLE_NAME);
qb.setProjectionMap(userProjectionMap);
//添加 where 条件,getPathSegments: 得到 uri 的 path 部分 content:XXX/user/1,
//get(1)得到 1
qb.appendWhere(UserData._ID+"="
                    +uri.getPathSegments().get(1));
SQLiteDatabase db=dh.getWritableDatabase();
//下面的 query 使用 qb 这个对象
Cursor c=qb.query(db, projection, selection, selectionArgs, null,
                        null, orderBy);
//也是通知下
/*
 * c.setNotificationUri(getContext().getContentResolver(), uri);
这里是把 Cursor C 添加到 ContentResolver 的监督对象组中去。
一旦有与 uri 相关的变化,ContentResolver 就会通知 Cursor C
```

```
*/
c.setNotificationUri(getContext().getContentResolver(), uri);
```

9.5 网络存储数据

前面介绍的几种存储都是将数据存储在本地设备上，除此之外，还有一种存储(获取)数据的方式，通过网络来实现数据的存储和获取。

可以调用 WebService 返回的数据或是解析 HTTP 协议实现网络数据交互。详细内容可以参见第 10 章网络应用。

下面是一个通过地区名称查询该地区的天气预报，以 POST 发送的方式发送请求到 webservicex.net 站点，访问 WebService.webservicex.net 站点上提供查询天气预报的服务。

代码如下：

```
import Java.util.ArrayList;
import Java.util.List;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.protocol.HTTP;
import org.apache.http.util.EntityUtils;
import Android.app.Activity;
import Android.os.Bundle;
public class WeatherActivity extends Activity {
    //定义需要获取的内容来源地址
    private static final String SERVER_URL=
        "http://www.webservicex.net/WeatherForecast.asmx
        /GetWeatherByPlaceName"; @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
         setContentView(R.layout.main);
        HttpPost request= new HttpPost(SERVER_URL);
        //根据内容来源地址创建一个 Http 请求添加一个变量
        List<NameValuePair> params= new ArrayList<NameValuePair>();
         //设置一个地区名称
         params.add(new BasicNameValuePair("PlaceName", "NewYork"));
         //添加必需的参数
        try {
            //设置参数的编码
```

```
                request.setEntity(new UrlEncodedFormEntity(params,
                                                HTTP.UTF_8));
                //发送请求并获取反馈
                HttpResponse httpResponse=
                        new DefaultHttpClient().execute(request);
                //解析返回的内容
                if(httpResponse.getStatusLine().getStatusCode() != 404){
                    String result=
                            EntityUtils.toString(httpResponse.getEntity());
                    System.out.println(result);
                }
            } catch(Exception e) {
                e.printStackTrace();
            }
        }
    }
```

别忘记了在配置文件中如下设置访问网络权限：

```
<uses-permission Android:name="Android.permission.INTERNET" />
```

9.6　实现方式总结

(1) 简单数据和配置信息，SharedPreference 是首选。

(2) 如果 SharedPreferences 不够用，那么就创建一个数据库。

(3) 结构化数据一定要创建数据库，虽然这稍显烦锁，但是好处无穷。

(4) 文件就是用来存储文件（也即非配置信息或结构化数据）的，如文本文件、二进制文件、PC 文件、多媒体文件、下载的文件等。

(5) 尽量不要创建文件。

(6) 如果创建文件，是私密文件或是重要文件，就存储在内部存储，否则放到外部存储。

第 10 章 网 络 应 用

Android 在网络通信方面也非常优秀，可以很轻松地使用 Android 自带的浏览器来访问网页。Android 基于 Linux 内核，它包含一组优秀的联网功能。目前，Android 平台有 3 种网络接口可以使用，它们分别是 Java. net. *（标准 Java 接口）、org. apache（Apache 接口）和 Android. net. *（Android 网络接口）。

Android SDK 中一些与网络有关的包如表 10-1 所示。

表 10-1 网络应用 jar 包

包	描 述
Java. net	提供与网络通信相关的类，包括流和数据包 socket、Internet 协议和常见的 HTTP 处理。该包是一个多功能网络资源。有经验的 Java 开发人员可以立即使用这个熟悉的包创建应用程序
Java. io	虽然没有提供现实网络通信功能，但是仍然非常重要。该包中的类由其他 Java 包中提供的 socket 和链接使用。它们还用于与本地文件的交互
Java. nio	包含表示特定数据类型的缓冲区的类。适用于两个基于 Java 语言的端点之间的通信
org. apache. *	表示许多为 HTTP 通信提供精确控制和功能的包。可以将 Apache 视为流行的开源 Web 服务器
Android. net	除核心 Java. net. * 类以外，包含额外的网络访问 socket。该包包括 URI 类，后者频繁用于 Android 应用程序开发，而不仅仅是传统的联网
Android. net. http	包含处理 SSL 证书的类

Android 与服务器通信的方式一般有两种。

(1) HTTP 通信方式：httpURLConnetction 接口、apache 的接口、httpClient 接口。HTTP 通信也分为 post 方式和 get 方式。

(2) Socket 通信方式：对于 TCP、Http、Socket、post、get 等概念和详细解释不属于本书的内容，大家可以阅读相关资料去了解。

10.1 Android 的 HTTP 通信

Android 除了兼容 J2ME 中的 Java. net api 外还提供了一些 Android 平台独有的 Android. net 包。功能更强大的是 org. apache. http 类，这个是 apache 实验室开源的包，对于 HTTP 请求处理很方便。

10.1.1 Java.net.HttpURLConnection 的 get 方式

下面实例说明如何使用 Get 方式。

```
String httpUrl= " http://www.sina.com/ ";
String resultData= "";
URL url= null;
try {
    //构造一个 URL 对象
    url= new URL(httpUrl);
} catch(MalformedURLException e) {
}
if(url != null) {
    try {
        //使用 HttpURLConnection 打开链接
        HttpURLConnection urlConn=
                (HttpURLConnection) url.openConnection();
        //得到读取的内容(流)
        InputStreamReader in=
                new InputStreamReader(urlConn.getInputStream());
        //为输出创建 BufferedReader
        BufferedReader buffer= new BufferedReader(in);
        String inputLine= null;
        //使用循环来读取获得的数据
        while(((inputLine= buffer.readLine()) != null)) {
            //在每一行后面加上一个"\n"来换行
            resultData+ = inputLine+ "\n";
        }
        //关闭 InputStreamReader
        in.close();
        //关闭 http 链接
        urlConn.disconnect();
    } catch(Exception e) {
    }
}
```

10.1.2 Java.net.HttpURLConnection 的 post 方式

下面实例说明如何使用 Post 方式。

```
String httpUrl= "http://192.168.0.1:8080/httpget.jsp";
//获得的数据
String resultData= "";
URL url= null;
try {                                   //构造一个 URL 对象
```

```
        url=new URL(httpUrl);
    } catch(MalformedURLException e) {
    }
    if(url != null) {
        try {
            //使用 HttpURLConnection 打开链接
            HttpURLConnection urlConn=
                        (HttpURLConnection) url.openConnection();
            //因为这个是 post 请求,设立需要设置为 true
            urlConn.setDoOutput(true);
            urlConn.setDoInput(true);
            //设置为 POST 方式
            urlConn.setRequestMethod("POST");
            //Post 请求不能使用缓存
            urlConn.setUseCaches(false);
            urlConn.setInstanceFollowRedirects(true);
        //配置本次连接的 Content-type,配置为 application/x-urlload 的
            //urlConn.setRequestProperty("Content-Type","application/x-urlload");
        //连接,从 postUrl.openConnection()至此的配置必须要在 connect
        //之前完成,要注意的是 connection.getOutputStream 会隐含地进行 connect
            urlConn.connect();
            //DataOutputStream 流
          DataOutputStream out=
                        new DataOutputStream(urlConn.getOutputStream());
            //要上传的参数
            String content="par="+URLEncoder.encode("ABCDEFG", "gb2312");
            //将要上传的内容写入流中
            out.writeBytes(content);
            //刷新、关闭
            out.flush();
            out.close();
            //获取数据
            BufferedReader reader=new BufferedReader(
                            new InputStreamReader(urlConn.getInputStream()));
            String inputLine=null;
            //使用循环来读取获得的数据
            while(((inputLine=reader.readLine()) != null)) {
                resultData+ = inputLine+ "\n";
            }
            reader.close();
            //关闭 http 链接
            urlConn.disconnect();
        } catch(Exception e) {
        }
```

```
}
```

10.1.3　org.apache.http 的 get 方式

下面实例说明如何使用 get 方式。

```
//http 地址
String httpUrl= "http://192.168.0.1:8080/httpget.jsp?par= Android_Get";
//HttpGet 链接对象
HttpGet httpRequest= new HttpGet(httpUrl);
try {
    //取得 HttpClient 对象
    HttpClient httpclient= new DefaultHttpClient();
    //请求 HttpClient,取得 HttpResponse
    HttpResponse httpResponse= httpclient.execute(httpRequest);
    //请求成功
    if(httpResponse.getStatusLine().getStatusCode()= = HttpStatus.SC_OK) {
        //取得返回的字符串
        String strResult= EntityUtils.toString(httpResponse.getEntity());
    }
} catch(Exception e) {
}
```

10.1.4　org.apache.http 的 post 方式

下面实例说明如何使用 post 方式。

```
//http 地址
String httpUrl= "http://192.168.0.1:8080/httpget.jsp";
//HttpPost 链接对象
HttpPost httpRequest= new HttpPost(httpUrl);
//使用 NameValuePair 来保存要传递的 Post 参数
List< NameValuePair> params= new ArrayList< NameValuePair> ();
//添加要传递的参数
params.add(new BasicNameValuePair("par", "HttpClient_Android_Post"));
try {
    //设置字符集
    HttpEntity httpentity= new UrlEncodedFormEntity(params, "gb2312");
    //请求 httpRequest
    httpRequest.setEntity(httpentity);
//取得默认的 HttpClient
    HttpClient httpclient= new DefaultHttpClient();
    //取得 HttpResponse
    HttpResponse httpResponse= httpclient.execute(httpRequest);
//HttpStatus.SC_OK 表示链接成功
```

```
        if(httpResponse.getStatusLine().getStatusCode()= = HttpStatus.SC_OK) {
            //取得返回的字符串
            String strResult= EntityUtils.toString(httpResponse.getEntity());
        }
    } catch(Exception e) {
}
```

10.2 设置代理

在手机联网的时候，优先选择 WIFI 网络，其次选择移动网络，这里所述移动网络主要是指 cmwap。大家都知道 cmwap 连接需要设置代理地址和端口。下面通过实例说明如何设置代理。

10.2.1 HttpURLConnection

使用中国移动的手机网络时，下面的方法可以直接获取 10.0.0.172,80 端口。

```
//通过 andorid.net.Proxy 可以获取默认的代理地址
String host= Android.net.Proxy.getDefaultHost();
//通过 andorid.net.Proxy 可以获取默认的代理端口
int port= Android.net.Proxy.getDefaultPort();
//定义代理
SocketAddress sa= new InetSocketAddress(host,port);
Proxy proxy= new Proxy(Java.net.Proxy.Type.HTTP,sa);
URL getUrl= new URL("www.baidu.com");
//设置代理
HttpURLConnection con= (HttpURLConnection) getUrl.openConnection(proxy);
```

10.2.2 HttpClient

```
DefaultHttpClient httpClient= new DefaultHttpClient();
//此处 Proxy 源自 Android.net
String host= Proxy.getDefaultHost();
int port= Proxy.getPort(context);
HttpHost httpHost = new HttpHost(host, port);
//设置代理
httpClient.getParams().setParameter(ConnRouteParams.DEFALLT_PROXY,
                                                        httpHost)
HttpGet httpGet= new HttpPost("< a href=
                                        "http://www.baidu.com"> www.baidu.com< /a> ");
HttpResponse response= httpClient.execute(httpGet);
```